1987-06

An interactive organizational choice processing system to support decision making by using a prescriptive garbage can model.

Kang, Sun Mo

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

AN INTERACTIVE ORGANIZATIONAL CHOICE
PROCESSING SYSTEM
TO SUPPORT DECISION MAKING BY USING
A PRESCRIPTIVE GARBAGE CAN MODEL

by

Kang, Sun Mo

June 1987

Thesis Advisor Taracad R. Sivasankaran

unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL *(If applicable)* 52	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS *(City, State, and ZIP Code)* Monterey, California 93943-5000	7b ADDRESS *(City, State, and ZIP Code)* Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL *(If applicable)*	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS *(City, State, and ZIP Code)*	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE *(Include Security Classification)* AN INTERACTIVE ORGANIZATIONAL CHOICE PROCESSING SYSTEM TO SUPPORT DECISION MAKING BY USING A PRESCRIPTIVE GARBAGE CAN MODEL

12 PERSONAL AUTHOR(S) Kang, Sun Mo

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT *(Year, Month, Day)*	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	1987 June	92

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	prescriptive garbage can model; organizational choice processing system

19 ABSTRACT *(Continue on reverse if necessary and identify by block number)*

This thesis discusses and implements an interactive decision support system using a Prescriptive Garbage Can Model. The fundamental presumption is that if the choice-outcome relationships in an organization can be observed and evaluated, it is possible to extract predictiveness from uncertain streams, and allow the organization to shift to a less random strategy. Solving organizational problems consists of selecting those choices that lead the organization in a direction towards the ideal state. Thus, it is convenient to model the organizational state transitions as a Markovian process with stationary properties. The purpose of a Prescriptive Garbage Can Model is to advise the participants of the choices available in a current situation, and to present choice policies leading the highest potential benefits. Also a method of interfacing the current system with an expert system for intelligent decision making is examined.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Taracad R. Sivasankaran	22b TELEPHONE *(Include Area Code)* (408) 646-2637	22c OFFICE SYMBOL Code 54Sj

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

unclassified

An Interactive Organizational Choice Processing System to support Decision Making by using A Prescriptive Garbage Can Model

by

Kang, Sun Mo
Major, Korean Army
B.S., Korean Military Academy, 1979

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1987

ABSTRACT

This thesis discusses and implements an interactive decision support system using a Prescriptive Garbage Can Model. The fundamental presumtion is that if the choice-outcome relationships in an organization can be observed and evaluated, it is possible to extract predictiveness from uncertain streams, and allow the organization to shift to a less random strategy. Solving organizational problems consists of selecting those choices that lead the organization in a direction towards the ideal state. Thus, it is convenient to model the organizational state transitions as a Markovian process with stationary properties. The purpose of a Prescriptive Garbage Can Model is to advise the participants of the choices available in a current situation, and to present choice policies leading the highest potential benefits. Also a method of interfacing the current system with an expert system for intelligent decision making is examined.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The Prescriptive Garbage Can Model (PGCM) of organizational decision-making [Refs. 1,2] can be defined as chance events resulting from the interactions of four elements in the organizational context, (i) problems, (ii) solutions, (iii) participants, and (iv) choice opportunities. As with every anarchic and random system, the participants desire to solve the current problem in the most effective manner. Which problems are actually taken up for action, in what priority, what choices are made in solving them, and how conclusively they are solved, are all functions of ambiguous preferences, and time and energy constraints of the participants.

A model imparting some degree of structure and comprehensibility to the complex organizational interactions and suggesting rational choice policies in an otherwise irrational context may be of invaluable assistance to organizational decision-makers. Thus, the model is prescriptive in nature. The building of such a model would link rational decision-making [Refs. 1,3] with anarchic decision-making [Ref. 2] thought.

Three objectives of the model are the following :

1. Advise the participants of the choices available to them in a specific organizational state
2. Estimate the expected benefit resulting from each choice
3. Lay down choice policies which would assist the participants in leading the organization in the long run to the state that has the highest potential benefits

Under severe lack of knowledge, decision makers may adopt a random search and choice rule, i.e., decisions are ill-defined, inconsistent, unclear, uncertain and problematic. Learning and outcomes are a matter of accidental trial-and-error.

While random strategies are always available, one may wonder whether they can be imbued with conscious thought processes to deal with uncertainty more effectively. If the choice-outcome relationships in an organization can be observed and evaluated, it is conceivable to extract predictiveness from uncertain streams, and thereby allow the organization to shift to a less uncertain strategy, in particular toward cybernetic and stochastic decision procedures.

This study discusses the design and implementation of the Prescriptive Garbage Can Model to provide a best course of actions on the anarchic organizational system.

Chapter II provides background on the prescriptive organizational model of garbage can choice policies. This includes a stochastic approach to the garbage can model, definitions and assumptions about the components of PGCM, and a prescriptive model of organizational choice. Chapter III examines the decision making process and discusses the design and implementation using a military offensive operation example. Chapter IV contains recommendations for further study on the topic. Appendix A is the source program. Appendix B is the user manual for the current implementation. Appendix C is a demonstration how offensive operation decision choices could be taken. Appendix D is a demonstration how university schedule decision choices could be taken.

II. BACKGROUND

A. A STOCHASTIC APPROACH TO THE PRESCRIPTIVE GARBAGE CAN MODEL

What appears on the surface as random organizational behavior is most likely not totally random, but casually influenced by a series of external factors and internal choices that can be modelled as probabilistic phenomena. It is often the difficulty of understanding numerous organizational and environmental forces that act simultaneously which renders probabilistic processes to appear as random occurrences. Thus, it may be useful to assume that organizations are ultimately more probabilistic in nature than purely random. The probabilistic approach obviously implies an inevitable degree of indeterminancy.

The prescriptive garbage can process, whereby problems, solutions, choices and participants are in organizational confluence, is made up of a large number of distinct actions sequenced over time. At any point in time, an organization can be characterized as belonging to a discrete organizational state. An organizational state is the conditional wherein essential characteristics of the organization (i.e., state variables) take on distinct and measurable values. During the fleeting existence of the organization in a specific state, if the participants were seeking globally optimal decisions, they would endeavor to identify the current state of the organization and exercise one of the choices that are available to them in that state. However, the effect of a decision may not be fully predictable. Thus, while a decision might be attractive in terms of an intended effect, an accurate decision calculus may not always be possible. Stated thus, organizational flux can be described as consisting of a stream of single-step state transitions over time due to the series of decisions made by the participants. In this perspective, stochastic modeling techniques may be applied to tame the transition phenomenon [Ref. 1].

Despite the probabilistic nature of the organization processes, organizational structures are ultimately considered to be homeostatic. This homeostasis concept relates to the capacity of the organization to withstand random perturbations which have not been foreseen by the participants [Ref. 1]. According to cyberneticians, an organization may be in any of the enormous number of possible states with related choice opportunities. Solving organizational problems consists of selecting those

choices that lead the organization in a direction towards the ideal state. Thus, it is convenient to model the organizational state transitions as a Markovian process with stationary properties. A process is stationary when organizational states become stable and invariant under time shifts. The homeostatic nature of the organizations implies the operation of at least some stationary properties.

B. DEFINITIONS AND ASSUMPTIONS

1. Organizational Elements

As defined in the PGCM, any organization consists of four relatively independent elements. They are (i) problems, (ii) solutions, (iii) participants, and (iv) choices. Relative independence implies that each element can assume its own identity, existence and relevance. In addition, we presume that problems are triggered by external or internal factors and represent the mismatch between the current organizational state and the desired state. Solutions are either tools or answers directly available within the organization waiting to be bound to the appropriate problems. Participants with their limited stocks of energy focus their attention on important problems and search for attractive solutions. Choices act as a cementing factor that ties the above three elements together.

2. Organizational States

The organizational state Z_i is a function of three attributes which describe an organization at a certain point in time. These attributes are :

1. The importance of the problems remaining to be solved (P_i),
2. The effectiveness of the solutions applied to problems (S_i) in the recent past,
3. The energy levels of the participants available for problem-solving (E_i).

The choice of P, S and E as attributes of organizational states is motivated by the structure of the PGCM which employs these elements as building blocks. P, S and E are assumed to be independent and measurable attributes. For convenience of representation, we shall use the coordinate system to denote a state. Thus an organizational state, $Z_i = (P_i , S_i , E_i)$.

3. Choices

Choices are decisions taken by participants in their pursuit to solve problems. They are determined by judging the nature of problems remaining to be solved, the effectiveness of the considered solutions, and the energy input available from the

participants required of a particular choice. In an organized anarchy, choices are assumed to be made accidentally. However, if choices were to be made rationally amidst the anarchy, they would presumably carry the organization towards the state (0,1,1). Rational managers would prefer such a state because they would like to see as many of the remaining important problems solved as possible, in an effective manner, and have at their disposal at all times a adequate supply of energy that can be applied to future problem solving. This is not to imply that managers wish to remain absorbed in state (0,1,1), since this means no opportunities, eternal calculations and unexpended energy. Rather, managers would prefer to attain a dynamic equilibrium at or close to (0,1,1). At such equilibrium, there is a continuous flow of problem opportunities and their effective resolution in a timely fashion so that sufficient manpower energy is readily available to meet new problem opportunities as soon as they arise.

In general, selecting a choice induces the transition of the organization to a new state in the next time interval. It is possible that taking no decisions is a choice in itself. It can shift the current state to a new state with more problems.

4. Choice Policies

Choice policies provide a prescriptive approach to problem solving. Once a set of organizational states and associated choices available therein can be identified, it is possible to bring to bear rationality in decision-making by laying down choice policies. Choice policies consist of suggestions as to what choices should be preffered while the organization is perceived to be in a particular state. In a sense, choice policies form a set of guidelines for organizational decision makers. Usually, the choice policies are so recommended as will most likely bring in the maximum benefits for the organization in the long run.

C. A PRESCRIPTIVE MODEL OF ORGANIZATIONAL CHOICE

1. Organizational Flux as Stochastic Transitions

Introducing rationality into an anarchic system requires that the decision-makers observe a calculus of outcomes based upon the (i) understanding of the implications of the various organizational states, (ii) knowledge of all the choices available to them in each state, and (iii) assessment of the probable impact of exercising a choice on the current state, before they reach a decision. We infuse rationality into the Prescriptive Garbage Can Model of anarchic actions through the use of a transition probability matrix.

The transition probability matrix represents the various organizational states, the available choices under each state, and the probabilities with which a choice can take the organization from one state to another. Z_i , i = 1 . . . . , n, denotes the organizational states; C_i (k), k = 1 , . . . , m_i , the choices available in a state i; q_{ij} c(k), the probability that the initial state Z_i will transit to Z_j when some choice C_i (k) is taken. Implicit in the matrix is the fact that there is no guarantee a choice can always lead to a state that is predictable beforehand. Impossible states may be filtered out from the matrix altogether and infeasible transitions may be represented by zeros. Note that $\sum_j q_{ij}\, c_i(k) = 1$. For simplicity of notation, we omit the subscript i in c_i (k), and denote by c(k).

The prescriptive model requires the determination of the transition probabilities. While several methods have appeared in the literature in estimating subjective probabilities, one that has evoked considerable interest in recent years consists of systematic elicitation of expert judgement [Refs. 1,4,5]. Expert knowledge and opinions often form an adequate surrogate, when historical data seem either inapplicable or unavailable.

The following steps describe the mechanics of generating the transition probability matrix :

- Step 1 : Determination of the set of organizational states, n.

First, determine the number of possible values p can take. For this divide the scale (0,1) into as many scale points as possible, say r. Assuming these scale points are uniformly distributed, the value of each scale point p^u can be generated using the formula,

$p^u = (u-1) / (r-1)$, where u = 1 ,..., r.

For example, if r=3, then $p^1 = 0$, $p^2 = 0.5$, $p^3 = 1$. The same formula can be applied to determine the scale points for S and E. The value of r need not have the same value for P, S, and E.

Second, generate all possible combinations of P^u , S^u , E^u to determine all organizational states. If r=2 for P, S and E, then the different organizational states can be described by one of the combinations, (P^1, S^1, E^1), (P^1, S^1, E^2), (P^1, S^2, E^1), (P^1, S^2, E^2), (P^2, S^1, E^1), (P^2, S^1, E^2), (P^2, S^2, E^1), and (P^2, S^2, E^2). In general, assuming the partitions are equal for P, S and E ($r_p = r_s = r_e$) the maximum number of possible organizational states that can be represented using the (P,S,E) coordinate

form is thus r^3. If $r=2$, these states can be denoted by $Z_i = (P_i, S_i, E_i)$ where $i = 1, \ldots, 8$. Thus, $Z_1=(P^1, S^1, E^1)$, $Z_2=(P^1, S^1, E^2)$, ..., $Z_8=(P^2, S^2, E^2)$.

Note that once each possible combination (P^u, S^u, E^u) is assigned to a specific state Z_i, $i=1,\ldots,n$, the actual values of P, S, E's in any state thereafter be refered to by P_i, S_i and E_i. The following Table 1 represents each organizational states.

TABLE 1

AN EXAMPLE ORGANIZATIONAL STATE

State Z_i	(P_i, S_i, E_i)	Remarks
1	(0.0,0.0,0.0)	
2	(0.0,0.0,1.0)	
3	(0.0,1.0,0.0)	
4	(0.0,1.0,1.0)	
5	(1.0,0.0,0.0)	
6	(1.0,0.0,1.0)	
7	(1.0,1.0,0.0)	
8	(1.0,1.0,1.0)	

- Step 2 : For each of the states Z_i , identify and filter all the conceivable and feasible choices.

Collect all these choices to form a set defined by $D_i = \sum_k c(k)$, where $k = 1, \ldots, m_i$. In complex organizations, exhaustive enumeration of choices may be a difficult task. However, it is not unrealistic for organizations to anticipate and equip themselves with as many available choices as they can to meet different possible situations.

- Step 3 : For an initial state Z_i , pick one of available choices

  As a result of c(k), assume the organization enters state Z_j.

- Step 4 : Estimate the probabilities P, S, E

$\underline{P}_{pi}\,'_{pj}\, c(k)$, where $j = 1, \ldots, n$ and $\sum_j \underline{P}_{pi}\,'_{pj}\, c(k) = 1$. Repeat for elements S and E. This gives $\underline{S}_{pi}\,'_{pj}\, c(k)$, and $\underline{E}_{pi}\,'_{pj}\, c(k)$.

- Step 5 : Compute the row of the transition probability matrix using the following formula

$$q_{ij}\, c(k) = \underline{P}_{pi\,,\,pj}\, c(k) * \underline{S}_{pi\,,\,pj}\, c(k) * \underline{E}_{pi\,,\,pj}\, c(k) \qquad \text{(eqn 2.1)}$$

herein, we notated $\underline{P}$, $\underline{S}$, $\underline{E}$ as estimation probabilities of P, S, E

- Step 6 : Repeat for all remaining (m_i - 1) choices in Z_i.

- Step 7 : Repeat steps 3-6 for the remaining (n - i) states.

The general layout of the transition probability matrix is shown Figure 2.1.

Next State

Current State						
	Z1, C(1)	q11C(1)	q12C(1)	q13C(1)	...	q1nC(1)
	Z1, C(2)	q11C(2)	q12C(2)	q13C(2)	...	q1nC(2)
	...				...	
	Z1, C(m1)	q11C(m1)	q12C(m1)	q13C(m1)	...	q1nC(m1)
	Z2, C(1)	q21C(1)	q22C(1)	q23C(1)	...	q2nC(1)
	Z2, C(2)	q21C(2)	q22C(2)	q23C(2)	...	q2nC(2)
	...				...	
	Z2, C(m2)	q21C(m2)	q22C(m2)	q23C(m2)	...	q2nC(m2)
	Z3, C(1)	q31C(1)	q32C(1)	q33C(1)	...	q3nC(1)
	Z3, C(2)	q31C(2)	q32C(2)	q33C(2)	...	q3nC(2)
	...				...	
	...				...	
	Z3, C(m3)	q31C(m3)	q32C(m3)	q33C(m3)	...	q3nC(m3)
	...				...	
	...				...	
	Zn, C(mi)	qn1C(mi)	qn2C(mi)	qn3C(mi)	...	qnnC(mi)

Number of states = Z(1, ... , n);

Number of choices in each state = C(1, ... , m_i);

Transition probability matrix satisfies the condition

$\sum_j q_{ij}\, C(k) = 1$, for k = 1, ... , m_i

Figure 2.1 An Example Transition Probability Matrix.

2. Goodness Measure of an Organizational State

For each organizational state Z_i , we assume there is an associated measure of goodness, g_i. This measure is ordinal in nature and reflects the amount of benefit derivable from the values of P, S and E corresponding to each state. The idea is similar to a balance sheet which conveys the state of health of an organization. S and E can be viewed as assets in a balance sheet, since they represent the strength of the organization. On the other hand, P can be viewed as a liability in that it detracts from the organizational performance. Note that high values of S_i and E_i imply high values of g_i. Conversely, high values of P_i imply low values of g_i. The composite amount of goodness for the state Z_i can be expressed as follows :

$$g_i = - P_i + S_i + E_i \qquad \text{(eqn 2.2)}$$

In theory, the ideal state of the organization corresponds to g = 2, since P = 0, S = 1, and E = 1. Contrarily, for the anti-ideal state, g = -1, since P = 1, S = 0 and E = 0. The following Table 2 shows each goodness measurement.

TABLE 2
AN EXAMPLE GOODNESS MEASUREMENT

State Z_i	(P_i , S_i , E_i)	Goodness	Remarks
1	(0.0,0.0,0.0)	0.0	
2	(0.0,0.0,1.0)	1.0	
3	(0.0,1.0,0.0)	1.0	
4	(0.0,1.0,1.0)	2.0	
5	(1.0,0.0,0.0)	-1.0	
6	(1.0,0.0,1.0)	0.0	
7	(1.0,1.0,0.0)	0.0	
8	(1.0,1.0,1.0)	1.0	

3. **Transition Benefit**

The goodness measure of an organizational state can be related to the transition probabilities through the idea of transition benefit. Transition benefit is the expected incremental goodness due to a transition that results from a specific choice. It is calculated as follows.

- Step 1 : Difference of goodness value between current state Z_i and terminal state Z_j for choice c(k)

$$(g_j - g_i)c(k) = -(P_jc(k) - P_i) + (S_jc(k) - S_i) + (E_jc(k) - E_i) \qquad \text{(eqn 2.3)}$$

- Step 2 : Expected incremental benefit (G) of the choice

$$G(Z_i, c(k)) = \sum_j (g_j - g_i)c(k) * q_{ij}c(k) \qquad \text{(eqn 2.4)}$$

If there are n states and $\sum_i m_i$ choices, the transition benefit matrix will be dimension of n x $\sum_i m_i$.

4. **Identification of a Choice Policy**

We have seen that policy is a prescriptive function. Its purpose is to suggest which choice c(k) out of the possible set of choices c(1,2, ... m_i) must be acted upon, given the organization is in state Z_i If rationality in decision making is assumed, choices will have to be so exercised as to maximize g_i. This can be achieved by maximizing the sum of the expected selection and sequencing of the different choices. Howard's algorithm can be employed to perform the maximization [Refs. 6,7]. The algorithm is applicable while dealing with a stochastic process where the law of transition and the corresponding benefit function are known. It consists of an intelligent trial and error iterative procedure that selects the best beneficial choice for each state in each iteration until the long run expected mean income per choice is maximized. The following one is the dynamic programming formulation.

$$V(S) = \max\{i(S,a) - g + \sum v(s)q_S \text{ , s } C(a)\}, \text{ for } s = 1, .., S \qquad \text{(eqn 2.5)}$$

Note S : initial state, s : next state, g : maximum mean income per period, a : chosen action, $q_{S,s}C(a)$: transition probability that transit from initial state S to next state s when action a is chosen.

5. Reinforcement of Choice Policies through Learning/Revision

From a cybernetic perspective, generating a choice policy is a learning process. The organization should continually examine the outcomes following from the choices it made in the previous periods, reinforce the assessments of the organizational elements P, S and E, and revise its battery of choices. This results in the re-evaluation of the transition probability matrix and consequently leads to a new set of choice policies for the next period.

III. SOFTWARE DESIGN AND IMPLEMENTATION

A. DECISION MAKING PROCESS

The Prescriptive Garbage Can Model refers to a class of systems which support the process of making decisions. The decision maker can retrieve data and test alternative solutions during the process of problem solving. This system also should provide ease of access to the data base containing relevant data and interactive testing of solutions. The system analyst must understand the process of decision making for each situation in order to analysis a system to support it. The model proposed by Herbert A. Simon consists of three major phases [Ref. 9], they are (i) intelligence phase, (ii) design phase, and (iii) choice phase.

1. Intelligence Phase

Searching the environment state calling for decisions. Estimation data are obtained, and examined for clues that may identify problems; set all estimation probabilities. One of the important fact is how to formulate the problems. A problem formulation might have a risk of solving the wrong problem, but the purpose of problem formulation is to clarify the problem so that design and choice activities operate on the right problem [Ref. 9]. Frequently, the process of clearly starting the problem is sufficient; in other cases, some reduction of complexity is needed. Four strategies for reducing complexity and formulating a manageable problem are [Ref. 9]

- Determining the boundaries
- Examining changes that may have precipitated the problem
- Factoring the problem into smaller subproblems
- Focusing on the controllable elements

A Prescriptive Garbage Can Model can obtain intelligence through searching, hence allow the user to approach the task heuristically through trial and error rather than by preestablished, fixed logical steps. So establishing analogy or relationship to some previously solved problem or class of problems is useful.

2. Design Phase

Inventing, developing, and analyzing possible courses of choices is performed in this phase. It involves processes to understand the problem, to generate solutions and to test solutions for feasibility. "A significant part of decision making is the

generation of alternatives to be considered in the choice phase" [Ref. 10]. The act of generating alternative is creativity that may be enhanced by alternative generation procedures and support mechanisms. In this process, an adequate knowledge of the problem area and its domain knowledge, and motivation to solve the problem will be required. Given these situations, analogies, brainstorming, checklists can enhance these creativities [Ref. 9].

3. Choice Phase

Selecting an choice from those available by using decision making software(.i.e., PGCM), can establish all choices for each organizational state.

B. DESIGNING THE PGCM HIERARCHY AND DFD

1. Hierarchical Program Structure

To process the PGCM, we first set up estimation probabilities and alternative actions through intelligence/design phase. Then we also establish choice policies through choice phase. Herein we focus on the choice phase that consists of two procedures. One is to produce all matrices such as organizational states, transition matrix, goodness measure table, benefit matrix from given user requirements specification. The other one is to apply these matrices to generate long run policy. Figure 3.1 shows the modules that are invoked by the main PGCM program. As we see, each of modules is a black box that takes input data, performs some transformation on that data, and process output data.

2. Data Flow Diagram

Since we are establishing modules as functional elements, we need to know what are the inputs/outputs. So for each module we will use a simple black box diagram to show the data flow of PGCM. For example, in Figure 3.2, there is a module labeled benefit matrix. Independent of all other modules in the program, we need transition matrix and goodness measure table. The followings are the inputs/outputs parameters for each module [Ref. 11].

** Module Getinfo (level 1.1) **

inputs : number of scale points for each factor (r_p , r_s , r_e)

number of different choices for each state

estimation probabilities

process: store input data via user interaction

output : estimation probability table

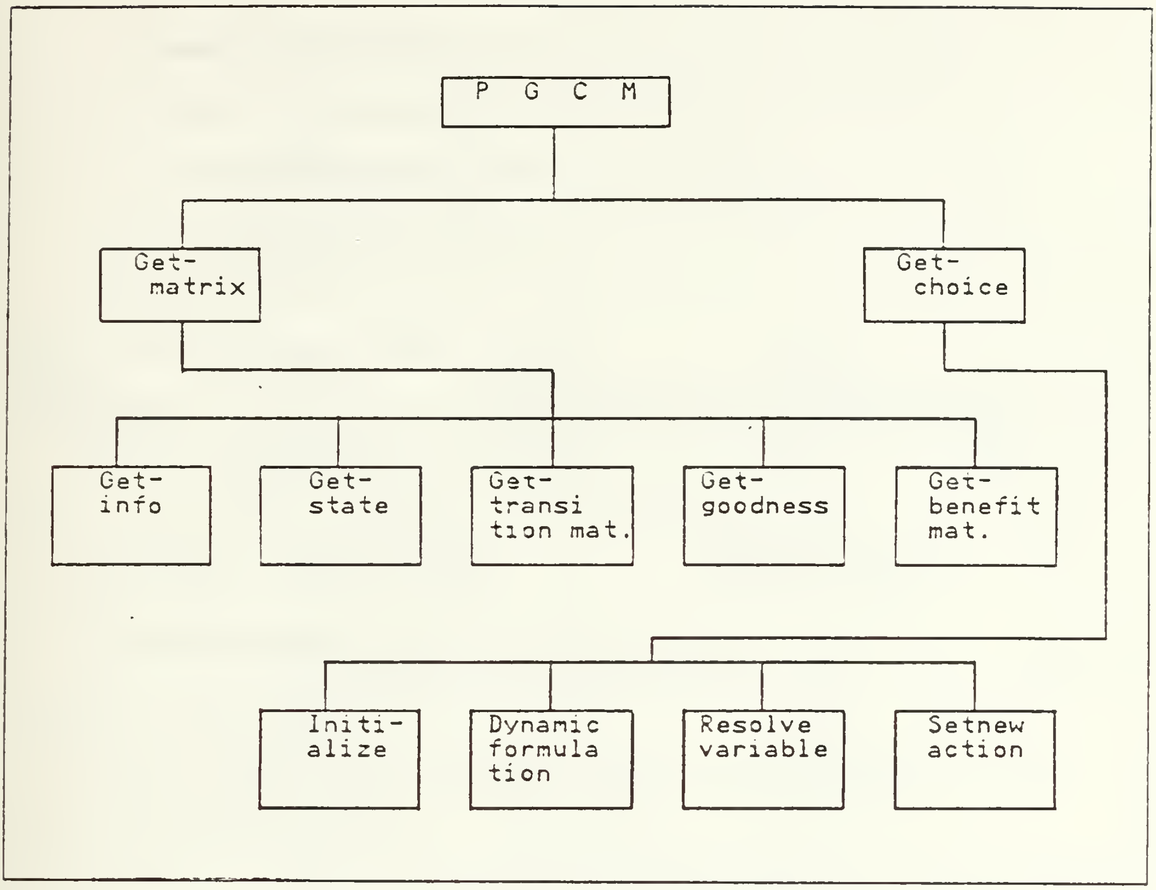


Figure 3.1 Hierarchical Program Structure for PGCM.

** Module Getstate (level 1.2) **

input : number of scale points for each factor (r_p , r_s , r_e)

process: combinate all scale points

output : organizational state table (size : $n = r_p * r_s * r_e$)

** Module Gettransition (level 1.3) **

inputs : estimation probabilities

organizational state table

process: calculate transition probability using formula 2.1

output : transition probability matrices (size : n * ($\sum m_i$ * n))

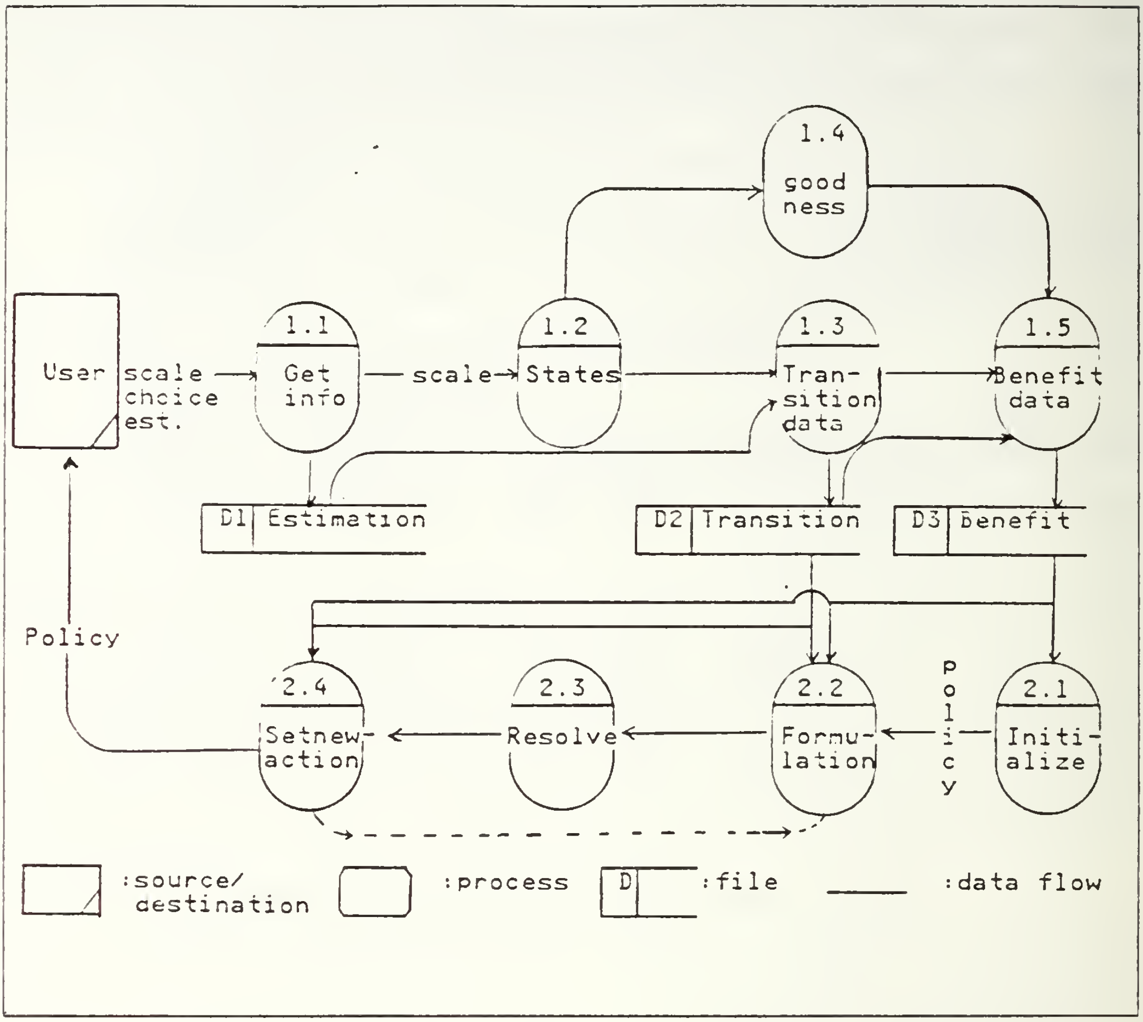


Figure 3.2 Data Flow Diagram for PGCM.

** Module Getgoodness (level 1.4) **

input : organizational state table

process: calculate goodness measure using formula 2.2

output : goodness measure table (size : n)

** Module Getbenefit (level 1.5) **

inputs : transition probability matrices

goodness measure table

process: calculate transition benefit probability using formula 2.3, 2.4

output : transition benefit matrix (size : n * maxchoice)

** Module Initialize (level 2.1) **

input : transition benefit matrix

process: select best choices for each state from transition benefit matrix that has the highest value in that state

output : policy table (size : n)

** Module Dynamicformulation (level 2.2) **

inputs : transition benefit matrix

transition probability matrices

temporary policy table

process: fill the coefficient table need to solve equations described by formula 2.5

output : coefficient table (size : n * n+1)

** Module Resolvevariable (level 2.3) **

input : coefficient table

process: resolve variable using gaussian elimination method

output : variable values

** Module Setnewaction (level 2.4) **

inputs : variable values

transition benefit matrix

transition probability matrices

process: set a new policy using howard algorithm

output : new policy table

C. PGCM PROCESS ALGORITHM

1. Input Data via Terminal

The current PGCM system needs to know number of scale points for each factor, different number of choices, and the estimation probability.

2. Generate Transition/Benefit Probability(formula 2.1,3,4)

3. Value Determination Operation

a) establish n linear simultaneous equations (v_i , g).

: use q_{ij} and i(S,a) for a given policy to solve

$g + v_i = i(S,a) + \sum q_{ij} v_j,\ i = 1, 2, ..., n$

b) set arbitary v_i equal to 0, normally v_n

c) resolve and produce the relative values using gaussian method.

4. Policy Improvement

a) find the alternative C(k) that maximizes the test quantity (v_i , g).

: find $\max\{i(S,a)\ C(k) + \sum q_{ij}\ C(k)\ v_j\}$ using the relative values v_i of the previous policy, then C(K) becomes the new decision in the ith state, i(S,a) C(K) becomes i(S,a), and q_{ij} C(K) becomes q_{ij}

b) perform this procedure for every state, and determine a new policy.

5. Combined Operation in An Iteration Cycle

a) select an initial policy from immediate benefit values.

b) solve the relative values v_i and g by setting v_n to 0.

c) find an alternative that has maximal benefit values.

d) if all alternatives are equally same benefit values, leave it unchanged

1) sort benefit values (1 .. choice(.n.))

2) calculate absolute difference for each benefit values

3) if a difference is less than 0.0001, take an old action else set a new action

e) repeat until the policies on two successive iterations are identical

f) if gain value g is decreased, then set arbitary v_{i-1} equal to 0, and repeat step3 thru step 5 until satisfied

The above algorithms step 3 thru step 5 based on the policy iteration method for multiple chain processes [Ref. 6].

D. IMPLEMENTATION WITH OFFENSIVE OPERATION EXAMPLE

Decision making in battle field involves unclear problems, chance event solutions, fluid energy derived from participants, and choices that seldom resolve problems. At any moment in time, a battle field may have a large number of problems to deal with, different possible solutions to cope with these problems, and many participants to make the necessary choices. Since taking into account all of the problems simultaneously could confuse the illustration, we shall assume that there is only one problem to be addressed during the interval of time considered and that the problem is related to offensive operation. We assume the representative elements in the offensive operation are number of attack forces, weapon systems, weather, relation to the consequent military operation. Figure 3.3 shows a system flow chart of the Prescriptive Garbage Can Model process.

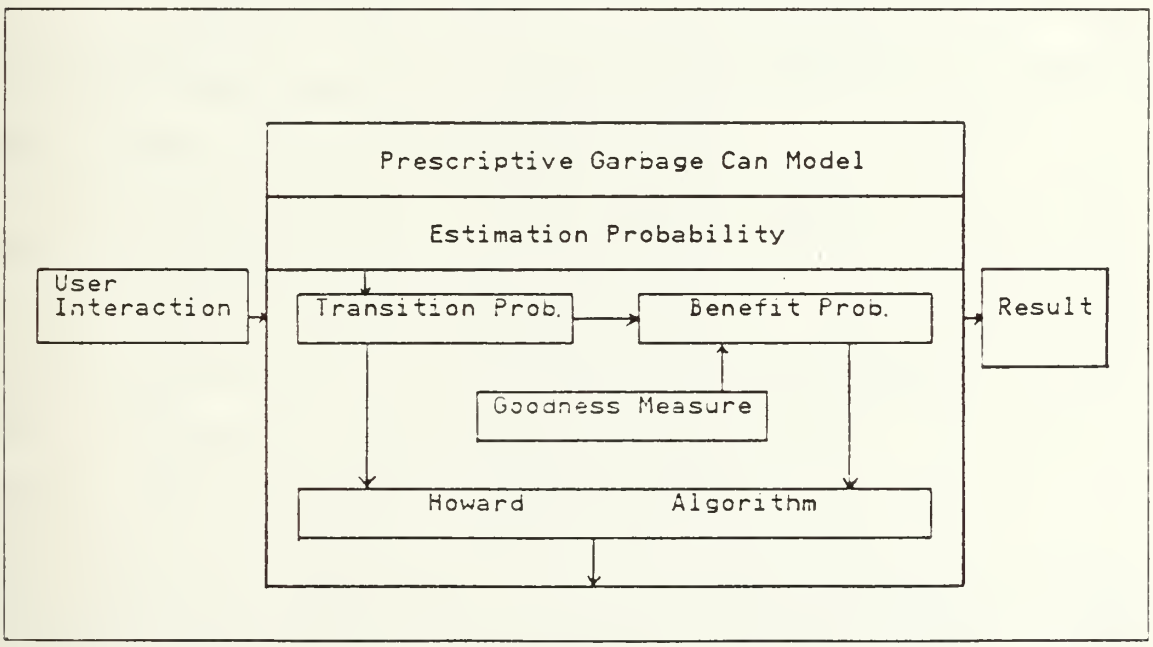


Figure 3.3 System Flow Chart of PGCM.

1. User Interaction

a. Define the problem and determine the organizational states

Once we define the problem and number of scale points for each factor, we can combinate the organizational states. We shall limit the number of scale points to P = 3 (0, 0.5, 1.0), S = 3 (0, 0.5, 1.0), E = 2 (0, 1.0). Thus, if we take into account all the combinations of (P_i , S_i , E_i), we can have at most 18(3x3x2) possible organizational states. Table 3 shows the description and the organizational states of this problem.

b. Identify and filter all the conceivable and feasible choices

In this example, we choose only 4 choices towards solving the offensive operation problem. They are

C(1) : Smoke operation and do attack

C(2) : Supporting high performance weapon systems

C(3) : Reinforcing attack forces

C(4) : Changing attack forces

For instance, (0.5, 0.5, 1.0) is one of these combinations denoted in our Table 3 by Z_{10}. It can correspond to a situation where an offensive forces have a good chance to attain their objectives successfully since they have no significant problems. But the military expert may have some questions such as the enemy's recovery ability from the previous shock or friendly forces's risks caused by a little shortage of attack forces, etc. So the military expert may look for another action to protect friendly forces such as a smoke operation. A smoke operation can significantly reduce the enemy's effectiveness in both the day time and at night. Combined with suppressive fire, smoke will provide increased opportunities for maneuver forces to deploy while minimizing losses. Also the effective delivery of smoke at the critical time and place on the battle field will contribute significantly to the combined arms team winning the first battle. Therefore, we we choose a smoke operation as one of feasible choices. The rest of the choices are also chosen in a same manner. Generally each state may have a different set of feasible choices for each state. For example, we can pick out C(4) since Z_6 is an ideal state we don't have to retain on C(4) which can be selected in the worst situation. so Z_6 has 3 choices. Hereby, we bring an interface problem between PGCM and expert system that identify and filter all the conceivable

TABLE 3

DESCRIPTION OF PROBLEMS AND ORGANIZATIONAL STATES

P : Importance of problems to be solved
S : Degree of effectiveness in problem-solving
E : Potential energy of participants

p1=0 No significant problem regarding attack forces, mission load, weather, relation with consequent military operation
p2=.5 Moderate shortage of attacking forces, not good weapon system good weather, a certain time delay to the consequent operation
p3=1 Acute shortage of attacking forces, big mission load, bad weather, a tremendous time delay to the consequent operation

S1=0 Most of personnel have no experience in the battle field, poor coordination with adjacent unit, poor performance weapon systems
S2=.5 Some personnel have an experience in the battle field, appropriate coordination and reasonable attack-defense forces ratio, good performance weapon systems, good logistic support systems
S3=1 Some personnel have an experience in the battle field, excellent coordination, best attack-defense forces ratio, excellent performance weapon systems , sufficient logistic support systems

E1=0 Not quite proud of their operations,passive action
E2=1 High morale, high responsibility

State Z_i	(P_i , S_i , E_i)	Remarks
1	(0.0,0.0,0.0)	
2	(0.0,0.0,1.0)	
3	(0.0,0.5,0.0)	
4	(0.0,0.5,1.0)	
5	(0.0,1.0,0.0)	
6	(0.0,1.0,1.0)	ideal
7	(0.5,0.0,0.0)	
8	(0.5,0.0,1.0)	
9	(0.5,0.5,0.0)	
10	(0.5,0.5,1.0)	
11	(0.5,1.0,0.0)	
12	(0.5,1.0,1.0)	
13	(1.0,0.0,0.0)	anti-ideal
14	(1.0,0.0,1.0)	
15	(1.0,0.5,0.0)	
16	(1.0,0.5,1.0)	
17	(1.0,1.0,0.0)	
18	(1.0,1.0,1.0)	

and feasible choices, if our problem has hundreds of organizational states. That system certainly helps all experts. For the simplicity, this example has all the same number of choices for each state, and Appendix D shows a different number of choices for each state.

c. *Estimate the probabilities using expert judgement*

Using expert judgement, estimate the probabilities $\underline{P}_{pi\ ,\ pj}\ c(k)$, where j = 1,2, ... , 18 and $\sum_j \underline{P}_{pi\ ,\ pj}\ c(k) = 1$. Repeat for elements S and E. This gives $\underline{S}_{pi\ \cdot\ pj}\ c(k)$, and $\underline{E}_{pi\ ,\ pj}\ c(k)$. Given $P_i = 0$ and C(1), P_i may transit to a new state P_j , where P_j can be 0, 0.5 or 1.0. By examining historical data and gathering advice from senior officers or military experts, assess the degree of influence of changing attacking forces on P_i and translate the assessment into matching probabilities, $P_{0\ ,\ 0}\ C(1)$, $P_{0\ ,\ 0\cdot 5}\ C(1)$, $P_{0\ ,\ 1}\ C(1)$, with which each of these three transitions can take place. In our Case, These probabilities are shown below.

$$\underline{P}_{pi\ ,\ pj}\ C(1)$$

Initial State $P_i = 0$ Choice C(1)		
Terminal State		
$P_j = 0$	$P_j = 0.5$	$P_j = 1.0$
0.98	0.01	0.01

The values shown in the table implies that if there are no major problems regarding offensive operation at the present time (since $P_i = 0$), then the chances are low that new problems might occur merely on account of changing attacking forces.

We now consider S_i , the effectiveness of solutions. Let $S_i = 0$ be the initial state of S_i. Focus on the same choice C(1). As before, we estimate the values of $S_{0\ \cdot\ 0}\ C(1)$, $S_{0\ \cdot\ 0\cdot 5}\ C(1)$, $S_{0\ \cdot\ 1\cdot 0}\ C(1)$. Their values in our case are shown below

$$\underline{S}_{pi\ \cdot\ pj}\ C(1)$$

Initial State $S_i = 0$ Choice C(1)		
Terminal State		
$S_j = 0$	$S_j = 0.5$	$S_j = 1.0$
0.97	0.02	0.01

We also consider E_i , the potential energy of participants that has only 2 scale points. Let $E_i = 0$ be the initial state of E_i. consider on the same choice C(1). Their values in our case are shown below

$$\underline{E}_{pi\ ,\ pj}\ C(1)$$

Initial State $S_i = 0$ Choice C(1)	
Terminal State	
$E_j = 0$	$E_j = 1.0$
0.95	0.05

2. **Transition Probability Matrix**

Use formula 2.1 to compute the row of the transition probability matrix, q_{1j} C(1), corresponding to state Z_1 and choice C(1). In this step, we consider all the three elements discussed above together to generate the joint transition probabilities, for the initial state Z_1 (0,0,0) of the battle field, due to changing attacking forces. Repeat to generate an transition matrix for the remaining 17 states. Table 4 shows the transition probability matrix of Z_1. The upper part of this table is an application of formula 2.1 in connection with changing attack forces and the lower part is for the remaining choices, C(2), C(3), and C(4). The remaining transition probability matrix, Z_2 , ... , Z_{18} will be shown in Appendix C.

3. **Goodness Measure**

The goodness measure for each state is computed using formula (2.2). The resulting g values for our example are given in Table 5. As shown in Table 5, state Z_6 represents the ideal state in that it yields the highest possible benefit, i.e., g=2. Conversely, the anti-ideal state, $Z_{13,}$ is the most adverse state for the organization since the corresponding benefit is the lowest.

4. **Transition Benefit Matrix**

Based on the transition probability matrix and the vector of goodness measure, compute the transition benefits using formulas 2.3 and 2.4.. The result of performing this procedure for all the initial states is shown in Table 6.

TABLE 4

TRANSITION PROBABILITIES IN Z_1

Computation of Transition Probabilities q_{1j} C(1)

Initial State: Z_{1}=(0,0,0); Choice C(1): Changing forces

Terminal State	Formula *P* *S* *E*	Transition prob.
(0.0,0.0,0.0)	.0,.0*.0,.0*.0,.0	0.98*0.97*0.95=0.903070
(0.0,0.0,1.0)	.0,.0*.0,.0*.0,1.	0.98*0.97*0.05=0.047530
(0.0,0.5,0.0)	.0,.0*.0,.5*.0,.0	0.98*0.02*0.95=0.018620
(0.0,0.5,1.0)	.0,.0*.0,.5*.0,1.	0.98*0.02*0.05=0.000980
(0.0,1.0,0.0)	.0,.0*.0,1.*.0,.0	0.98*0.01*0.95=0.009310
(0.0,1.0,1.0)	.0,.0*.0,1.*.0,1.	0.98*0.01*0.05=0.000490
(0.5,0.0,0.0)	.0,.5*.0,.0*.0,.0	0.01*0.97*0.95=0.009215
(0.5,0.0,1.0)	.0,.5*.0,.0*.0,1.	0.01*0.97*0.05=0.000485
(0.5,0.5,0.0)	.0,.5*.0,.5*.0,.0	0.01*0.02*0.95=0.000190
(0.5,0.5,1.0)	.0,.5*.0,.5*.0,1.	0.01*0.02*0.05=0.000010
(0.5,1.0,0.0)	.0,.5*.0,1.*.0,.0	0.01*0.01*0.95=0.000095
(0.5,1.0,1.0)	.0,.5*.0,1.*.0,1.	0.01*0.01*0.05=0.000005
(1.0,0.0,0.0)	.0,1.*.0,.0*.0,.0	0.01*0.97*0.95=0.009215
(1.0,0.0,1.0)	.0,1.*.0,.0*.0,1.	0.01*0.97*0.05=0.000485
(1.0,0.5,0.0)	.0,1.*.0,.5*.0,.0	0.01*0.02*0.95=0.000190
(1.0,0.5,1.0)	.0,1.*.0,.5*.0,1.	0.01*0.02*0.05=0.000010
(1.0,1.0,0.0)	.0,1.*.0,1.*.0,.0	0.01*0.01*0.95=0.000095
(1.0,1.0,1.0)	.0,1.*.0,1.*.0,1.	0.01*0.01*0.05=0.000005

Transition Probabilities q_{1j} c(k)

State Z_1 q_{1j} C(k) for j = 1, ... , 18; k = 1, ... , 4

state	C(1)	C(2)	C(3)	C(4)
1	0.903070	0.039200	0.617400	0.018000
2	0.047530	0.156800	0.264600	0.072000
3	0.018620	0.137200	0.061740	0.036000
4	0.000980	0.548800	0.026460	0.144000
5	0.009310	0.019600	0.006860	0.006000
6	0.000490	0.078400	0.002940	0.024000
7	0.009215	0.000400	0.006300	0.036000
8	0.000485	0.001600	0.002700	0.144000
9	0.000190	0.001400	0.000630	0.072000
10	0.000010	0.005600	0.000270	0.288000
11	0.000095	0.000200	0.000070	0.012000
12	0.000005	0.000800	0.000030	0.048000
13	0.009215	0.000400	0.006300	0.006000
14	0.000485	0.001600	0.002700	0.024000
15	0.000190	0.001400	0.000630	0.012000
16	0.000010	0.005600	0.000270	0.048000
17	0.000095	0.000200	0.000070	0.002000
18	0.000005	0.000800	0.000030	0.008000

TABLE 5

EVALUATING GOODNESS MEASURES

State (Z_i)	(P_i	, S_i	, E_i)	g_i	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	1.0	1.0	
3	0.0	0.5	0.0	0.5	
4	0.0	0.5	1.0	1.5	
5	0.0	1.0	0.0	1.0	
6	0.0	1.0	1.0	2.0	ideal
7	0.5	0.0	0.0	-0.5	
8	0.5	0.0	1.0	0.5	
9	0.5	0.5	0.0	0.0	
10	0.5	0.5	1.0	1.0	
11	0.5	1.0	0.0	0.5	
12	0.5	1.0	1.0	1.5	
13	1.0	0.0	0.0	-1.0	anti-ideal
14	1.0	0.0	1.0	0.0	
15	1.0	0.5	0.0	-0.5	
16	1.0	0.5	1.0	0.5	
17	1.0	1.0	0.0	0.0	
18	1.0	1.0	1.0	1.0	

TABLE 6

TRANSITION BENEFIT MATRIX OF ALL CHOICES

State Z_i	Choices C(k)			
State	C(1)	C(2)	C(3)	C(4)
1	0.055000	1.235000	0.340000	0.800000
2	-0.005000	0.425000	-0.010000	-0.300000
3	0.175000	1.180000	0.580000	0.400000
4	0.115000	0.370000	0.230000	-0.700000
5	0.020000	0.770000	0.270000	-0.100000
6	-0.040000	-0.040000	-0.080000	-1.200000
7	0.160000	1.595000	0.750000	1.200000
8	0.100000	0.785000	0.400000	0.100000
9	0.280000	1.540000	0.990000	0.800000
10	0.220000	0.730000	0.640000	-0.300000
11	0.125000	1.130000	0.680000	0.300000
12	0.065000	0.320000	0.330000	-0.800000
13	0.090000	1.600000	1.045000	1.700000
14	0.030000	0.790000	0.695000	0.600000
15	0.210000	1.545000	1.285000	1.300000
16	0.150000	0.735000	0.935000	0.200000
17	0.055000	1.135000	0.975000	0.800000
18	-0.005000	0.325000	0.625000	-0.300000

5. **Generate the long run choice policy**

The objective of this step is to determine the offensive operation policy by evaluating what choices result in highest benefits in the long run as the battle field stochastically transits from one state to another. The mathematics of maximization of the long run benefit, when the law of transition shown in Table 4 and benefit function shown in Table 6 are known, can be achieved using Howard's algorithm. Let's trace the long run choice policy by applying formula 2.4.

a. Initialize policy table

Select best choices for each state from the benefit matrix that has maximum value among C(1) thru C(5), and set up a policy table as the following table.

Origin Choice Policy Table
S1 S2 S3 . . S7 S8 S9 S10 . . . S15 S16 S17 S18
2 2 2 . . 2 2 2 2 . . . 2 3 2 3

b. Resolve variables and evaluate max property

Once we have resolved all the variable values (v(1), ... , v(18), g) using gaussian elimination method [Ref. 8] we check to see if these resolved values satisfy the maximal property expressed in formula 2.4. For each state-action pair S,a, we evaluate $i(S,a) - g + \sum v(s)q_S$, s C(a) and then for each state S choose the maximizing act "a" [Ref. 7]. This leads to

State i(S,a) $\sum v(s)q_S$, s C(a)

(1,1) 0.055-g+0.903070v(1)+0.047530v(2) .. = 1.00E+00 - g

(1,2) 1.235-g+0.039200v(1)+0.156800v(2) .. = 1.00E+00 - g*

(1,3) 0.340-g+0.617400v(1)+0.264600v(2) .. = 1.00E+00 - g

(1,4) 0.800-g+0.018000v(1)+0.072000v(2) .. = 1.00E+00 - g

... ...

... ...

... ...

(10,1) 0.220-g+0.000019v(2)+0.001881v(3) .. = -6.25E-16 - g*

(10,2) 0.730-g+0.000070v(2)+0.006930v(3) .. = -1.50E-15 - g

(10,3) 0.640-g+0.000400v(2)+0.007600v(3) .. =-1.62E-15 - g
(10,4) -0.300-g+0.006000v(2)+0.014000v(3) .. =-8.60E-16 - g

... ...
... ...
... ...

c. Set a new policy table

At the above step, we observed a new policy, marked by the asterisks and they are

New Choice Policy Table
S1 S2 S3 . . . S7 S8 S9 S10 . . . S15 S16 S17 S18
2 4 2 . . . 2 2 2 1 . . . 2 3 2 1

d. Compare a new policy table with a previous one

Repeat b) and c) until a new policy table and a previous policy table correspond each other or the maximal income per unit time (g) is decreased. The latter case ,occasionally happens, and is possible in the case of g value near zero. For this unstable state, we set recursively arbitary V_{i-1} equal to 0, and repeat a) thru c).

e. Test result

As a result of prescriptive garbage can model execution, we lay down a policy table shown in Table 7 that have solved expressed by formula 2.5. Given Table 7 recommend and advise the commander of the best choices available in a specific organizational state. For example, if given situations are (i) Acute shortage of attacking forces, big mission load, a tremendous time delay to the consequent operation, (ii) Most of personnel have no experience in the battle field, poor coordination with adjacent, poor performance weapon systems, (iii) Not quite proud of their operations, we would like to change attack forces known as choice 4 to achieve his goal. Like this we can select choice 1 (do smoke operation) in state 6,10,18, choice

2 (supporting high performance weapon systems) in state 1,3,4,7,8,9,11,14,15,17, choice 3 (reinforcing attack forces) in state 12,16, and choice 4(changing attack forces) in state 2,5,13.

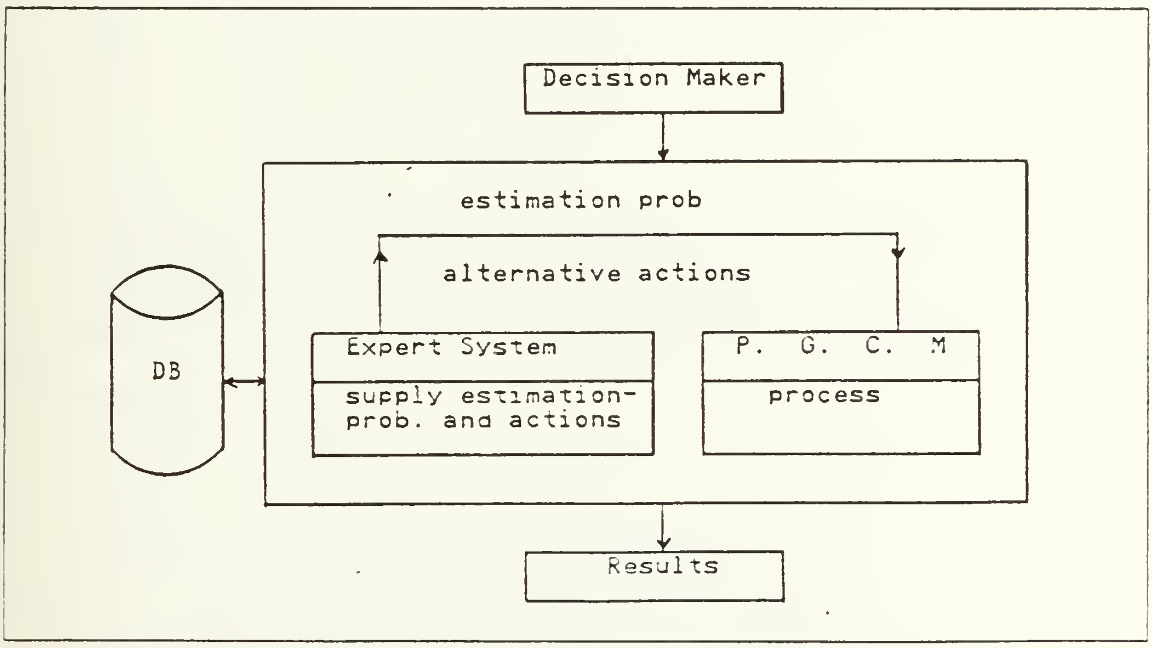
TABLE 7
SELECTED CHOICE POLICIES

State Z_i	Choice Policy C(k)
1	2
2	4
3	2
4	2
5	4
6	1
7	2
8	2
9	2
10	1
11	2
12	3
13	4
14	2
15	2
16	3
17	2
18	1

IV. FURTHER RECOMMENDED STUDIES

This thesis considers a prescriptive garbage can model to advise the participants of the choices available to them in a specific organizational state, and implements it to generate a choice policy table. The current system covers chance events resulting from the interactions of four elements in the organizational context, (i) problems, (ii) solutions, (iii) participants, and (iv) choice opportunities. The computer program could be modified to process a greater number of organizational states depend on a memory allocation(current system maxchoice 5, maxstate 36).

An ideal PGCM would be one that interfaces with an expert system to automatically transfer estimation probabilities and feasible actions about the organization problem into PGCM system without human intervention. The following diagram shows how an expert system would be used.



Interface between Expert System and PGCM

An expert system uses methods of reasoning to eliminate bad courses of actions and to determine the best courses of actions to achieve a goal. Expert systems use information in an intelligent way to perform tasks that are normally associated with

human experts. There are many anarchic and random situations, but human experts have some difficulties to find the best choice every time. Hereby we illustrated a diagram as one possible model to interface between PGCM system and expert system to determine the best actions for the given set of organizational states. The next step in this study is how to interface an expert system to estimate transition probabilities and feasible actions for input to the prescriptive garbage can model.

APPENDIX A

A SOURCE PROGRAM

```
PROGRAM  PGCMPROG(Input, Output);
(*$s 400000 *)
(**********************************************************
 **  TITLE         : PRESCRIPTIVE GARBAGE CAN MODEL       **
 **  AUTHOR        : Maj Kang, Sun Mo                     **
 **  Date Written  : 11 Feb - 19 May 87                   **
 **                                                       **
 **  Product       : Version 1.                           **
 **  System Used   : IBM 3033 VM/CMS                      **
 **                                                       **
 **  I/O Process   : Terminal Keyboard                    **
 **                                                       **
 **  Description   : This program is an interactive       **
 **                  choice processing system to support  **
 **                  decision making by using Prescriptive**
 **                  Garbage Can Model                    **
 **********************************************************)

(** Global Constants **)
const
     zero       = 0;
     one        = 1;
     two        = 2;
     three      = 3;
     four       = 4;
     six        = 6;
     eight      = 8;
     ten        = 10;
     seventy    = 70;
     maxfactor = 3;  (* number of variable                            *)
     maxchoice = 5;  (* number of maximum choice for each state       *)
     maxscales = 5;  (* number of maximum scale point for one factor *)
     maxstate  = 36; (* number of maximum states                      *)
     maxstateetc=37; (* number of maximum states plus one             *)
     maxrow    = 180;(* number of maxsrows, max-state*choice          *)
type
     counter    = 0..maxint;
     inputtype = record
       line      : array(.one..seventy.) of char;
       length    : counter;
       last      : counter;
     end;
     commands  = (EXECGCM, EXECEXIT, BAD);
     usermsgs  = (BADLINE, NOINPUT, NOINTERACT, ETCVAL, ETCCHO,
                    NONNUM, OVERSTATE, IMPOSSIBLE);
     choicetable   = array(.one..maxstate.) of integer;
     transitiontable=array(.one..maxrow,    one..maxstate.) of real;
     benefittable  = array(.one..maxstate, one..maxchoice.) of real;
     policytable   = array(.one..maxstate.) of integer;

(** Global Variables **)
var
     possiblestates,
     int, vi          : integer;
     gain             : real;
     wait             : char;
     choices    : choicetable;
     tranmatrix: transitiontable;
     benematrix: benefittable;
```

```
policy    : policytable;
userinp   : inputtype;
command   : commands;
helpfile  : text;
quitnow,
goodvalue : boolean;
```

```
(* ########################################### *)
(* ###                                     ### *)
(* ###   G  C  M           PART   I.       ### *)
(* ###          Functions                  ### *)
(* ########################################### *)
(* =========================================== *)
(* ===   Function      Getinp              === *)
(* =========================================== *)
Function getinp(var userinp : inputtype):boolean;
(* Get a single-letter command,
    making sure it is in the set of valid commands *)
var
    ch     : char;
begin
  userinp.length := 0;
  userinp.last   := 0;
  if eof then getinp := false
    else begin
       while not eoln do begin
          read(ch);
          if userinp.length < seventy then begin
              userinp.length := userinp.length + 1;
              userinp.line(.userinp.length.) := ch;
          end
       end;
       readln;
       getinp := true;
    end;
end; (*getinp *)

(* =========================================== *)
(* ===   Function      Skipblanks          === *)
(* =========================================== *)
Function skipblanks(var userinp : inputtype):boolean;
var
    blank  : boolean;
begin
  blank := true;
  while (userinp.last < userinp.length) and blank do begin
    userinp.last := userinp.last + 1;
    if userinp.line(.userinp.last.) <> ' ' then
      blank := false
  end;
  if not blank then
    userinp.last := userinp.last - 1;
  skipblanks := blank;
end; (* skipblanks *)

(* =========================================== *)
(* ===   Procedure     Getchar             === *)
(* =========================================== *)
procedure getchar(var userinp:inputtype; var ch:char);
begin
  if userinp.last < userinp.length then begin
     userinp.last := userinp.last + 1;
     ch := userinp.line(.userinp.last.);
  end else
    ch := ' ';
end; (* get char *)

(* =========================================== *)
(* ===   Procedure     Writeuser           === *)
(* =========================================== *)
procedure writeuser(msg:usermsgs);
begin
  case msg of
```

```
    BADLINE    : writeln('Bad Input, try again       <press enter key>');
    NOINPUT    : writeln('Have no data, try again    <press enter key>');
    NOINTERACT: writeln('Nothing typed, try again    <press enter key>');
    NONNUM     : writeln('Nonnumeric data, try again <press enter key>');
    ETCVAL     : begin
                     write  ('Available scale point : 2 to 9, try again ');
                     writeln(  '<press enter key>');
                 end;
    ETCCHO     : begin
                     write  ('Available choice : 2 to 5, try again   ');
                     writeln('<press enter key>');
                 end;
    OVERSTATE : begin
                     write  ('Maximum organizational states is less ');
                     writeln('than 36, try again   <press enter key>');
                 end;
    IMPOSSIBLE: begin
                     write  ('Cannot set up, try again with new data  ');
                     writeln('<press enter key>');
                 end;
  end;
  readln(wait);
end;

(********************************************)
(*****   Function      Getcommand        *****)
(********************************************)

Function getcommand(var command:commands):boolean;
(* Get a single-letter command,
    make sure it is in the set of valid commands *)
var
    ch      : char;
    userinp: inputtype;
begin
  page;
  writeln('****************************************************':67);
  writeln('***                                              ***':67);
  writeln('*** GCM Program Options  are the followings      ***':67);
  writeln('*** --------------------------------------       ***':67);
  writeln('***                                              ***':67);
  writeln('***                                              ***':67);
  writeln('***  1. ExecGCM (Execute GCM Program       )     ***':67);
  writeln('***                                              ***':67);
  writeln('***  2. ExecExit(Execution stop            )     ***':67);
  writeln('***                                              ***':67);
  writeln('***  =====> Type, Number   !!!!                  ***':67);
  writeln('****************************************************':67);
  getcommand := false;
  command := BAD;
  if getinp(userinp) then begin
     getcommand := true;
     if not skipblanks(userinp) then begin
        getchar(userinp,ch);
        if skipblanks(userinp) then
          if ch in (.'1','2'.) then
            case ch of
              '1' : command := EXECGCM ;
              '2' : command := EXECEXIT;
            end
          else getcommand := false;
     end;
  end;
end;
```

```
(* ########################################### *)
(* ###                                     ### *)
(* ###   G  C  M          PART    II.      ### *)
(* ###        Generate  Matrix             ### *)
(* ###                                     ### *)
(* ### generate transition probability,    ### *)
(* ### transition benefit probability,     ### *)
(* ### organizational states, goodness     ### *)
(* ### measure table: Input for Part III   ### *)
(* ########################################### *)

(*********************************************)
(*****   GET TRANSITION, BENEFIT MATRIX   *****)
(*********************************************)

Procedure Getmatrix(var possiblestates:integer; var choices:choicetable;
            var tranmatrix:transitiontable; var benematrix:benefittable);
const
    maxirow        = 45;
type
    scaletable     = array (.one..maxfactor.) of integer;
    scalevaltable  = array (.one..maxscales, one..maxscales.) of real;

    estimationtable=array(.one..maxfactor,one..maxirow,one..maxscales.)
                                         of real;
    combination    = record
      column : array (.one..maxfactor.) of real;
     end;
    statematrix    = array (.one..maxstate.) of combination;
    goodnessmatrix = array (.one..maxstate.) of real;
var
    numfactor,
    maxcho          : integer;
    scales          : scaletable;
    scalevals       : scalevaltable;
    statmatrix      : statematrix;
    goodmatrix      : goodnessmatrix;
    estmatrix       : estimationtable;
(* ***************************************** *)
(* ***   Get Number of Factors           *** *)
(* ***************************************** *)

Procedure getnumoffactor(var numfactor:integer;
                          var possiblestates:integer;
                          var scales:scaletable);
var
    i         : integer;
    ch        : char;
    indomain  : boolean;

(* ========================================== *)
(* ===   Function  Get Max # of States    === *)
(* ========================================== *)

Function  Getmaxnum(num : integer; scales : scaletable):integer;
var
    base, i : integer;
begin
   base     := one;
   for i := 1 to num do
    base := base * scales(.i.);
   getmaxnum := base;
end;

begin
    writeln('Number of factors : 3 -- P, S, E');
    writeln('________________________________');
    writeln;
    numfactor := 3;

    writeln('Enter # of scale points for each factor : ');
    writeln('__________________________________________');
```

```
     writeln;
     (********** get number of scale point **************)
     indomain := false;
     repeat (*repeat 1*)
     i := 1;
     repeat (*repeat 2*)
       goodvalue := false;
       writeln('factor',i:2, ' : ');
       if getinp(userinp) then
         begin
           if not skipblanks(userinp) then
             begin
               getchar(userinp,ch);
               if skipblanks(userinp) then
                 if ch in (.'2','3','4','5','6','7','8','9'.) then
                   begin
                     case ch of
                      '2' : int := 2;
                      '3' : int := 3;
                      '4' : int := 4;
                      '5' : int := 5;
                      '6' : int := 6;
                      '7' : int := 7;
                      '8' : int := 8;
                      '9' : int := 9;
                     end; (*caseend*)
                     goodvalue := true;
                   end (*endif ch*)
                 else writeuser(ETCVAL) (*else ch*)
               else writeuser(ETCVAL) (*else shipblanks*)
           end (*endif notskipblanks*)
           else writeuser(NOINPUT) (*else notskipblanks*)
         end  (*endif getinp*)
       else writeuser(NOINTERACT); (*else getinp*)
       if goodvalue then
         begin
           scales(.i.) := int;
           i := i + 1;
         end
     until (i = numfactor+one); (*end repeat2 *)
     (********** check and correct scale point **********)
     repeat (*repeat3*)
       writeln('# of scale points are : ');
       for i := one to numfactor do
         writeln('factor',i:2,' : ',scales(.i.));
       writeln('goahead : Press any key, correction : "%"');
       writeln; writeln;
       readln(ch);
       if ch = '%' then begin
         writeln('Enter factor number and value : ');
         readln(i, scales(.i.)); end;
     until not(ch = '%'); (*end repeat3*)
     possiblestates := getmaxnum(numfactor, scales);
     indomain := (possiblestates <= 36);
     if not indomain then writeuser(OVERSTATE);
 until indomain; (*end repeat 1*)
end;

(* ***************************************** *)
(* ***   Get Number of Choices          *** *)
(* ***************************************** *)
Procedure getnumofchoices(numfactor:integer; possiblestates:integer;
                          var choices:choicetable; var maxcho:integer);
var
     i          : integer;
     ch         : char;
begin
```

```
    write  ('Get Number of Choices for each State Z(i)   ');
    writeln('i = 1 ..',possiblestates:3);
    write  ('____________________________________________');
    writeln('__________'); writeln;
    (********** get number of choice  **************)
    i := 1;
    repeat
      goodvalue := false;
      writeln('State',i:3, ' : ');
      if getinp(userinp) then
        begin
          if not skipblanks(userinp) then
            begin
              getchar(userinp,ch);
              if skipblanks(userinp) then
                if ch in (.'2','3','4','5'.) then
                  begin
                    case ch of
                      '2' : int := 2;
                      '3' : int := 3;
                      '4' : int := 4;
                      '5' : int := 5;
                    end; (*caseend*)
                    goodvalue := true;
                  end (*endif ch*)
                else writeuser(ETCCHO) (*else ch*)
              else writeuser(ETCCHO) (*else shipblanks*)
          end (*endif notskipblanks*)
          else writeuser(NOINPUT) (*else notskipblanks*)
        end  (*endif getinp*)
      else writeuser(NOINTERACT); (*else getinp*)
      if goodvalue then
        begin
          choices(.i.) := int;
          i := i + 1;
        end
    until (i = possiblestates+one); (*end repeat2 *)
    (********** check and correct     **************)
    writeln('State Z(i)       # of choices');
    writeln('__________       ____________');
    repeat
      for i := one to possiblestates do
        writeln(i:6, choices(.i.):20);
      writeln('goahead : Press any key, correction : "%"');
      writeln; writeln;
      readln(ch);
      if ch = '%' then begin
        writeln('Enter state  number and value : ');
        readln(i, choices(.i.)); end;
    until not(ch = '%');
    maxcho := -999;
    for i := one to possiblestates do
      if choices(.i.) > maxcho then maxcho := choices(.i.);
end;

(* ***************************************** *)
(* ***   Get Estimate Probability        *** *)
(* ***************************************** *)
Procedure getestprob(numfactor:integer; maxcho:integer;
                         scales:scaletable; var scalevals:scalevaltable;
                     var estmatrix:estimationtable);
var
    i,j,k,l,m,n : integer;
    p           : real;
    ch          : char;
(* ========================================= *)
(* ===   Get scale points of each factor  === *)
```

```
(* ========================================= *)
Procedure Getvalofscale(num:integer; var scalevals:scalevaltable);
var
    i,j,k    : integer;
    p        : real;
begin
   for i := 1 to num do
     for j := 1 to scales(.i.) do begin
        p := (j-1)/(scales(.i.)-1);
        k := round(p * 100);
        scalevals(.i,j.) := k/100;
     end;
end;
begin
    getvalofscale(numfactor,scalevals);
    (********** Get estimation probability ************)
    writeln('Get Estimation Probabilities');
    writeln('____________________________');
    writeln;
    for i := one to maxcho do
      for j := one to numfactor do
        begin
          writeln('(factor',j:1,')');
          for k := one to scales(.j.) do
            begin
              write  ('Initial  State f(',j:1,') = ');
              writeln(scalevals(.j,k.):6:2,'  Choice  C(',i:1,')');
              writeln('Terminal State : ');
              for l := one to scales(.j.) do
                write  (' prob',l:1,' '); writeln;
              (***** read estimation probability *****)
              for l := one to scales(.j.) do
                read (estmatrix(.j,(i-1)*scales(.j.)+l,k.));
              readln;
            end; (*end k*) writeln; writeln;
          (********** check and correct *************)
          repeat
            write(' ':10);
            for k := one to scales(.j.) do
              write  ('-------'); writeln;
            write(' ':10);
            for k := one to scales(.j.) do
              write  (scalevals(.j,k.):6:2, ' '); writeln;
            write(' ':10);
            for k := one to scales(.j.) do
              write  ('-------');writeln;
            for k := one to scales(.j.) do
              begin
                write(' ':5,i:1,' ':4);
                for l := one to scales(.j.) do
                write(estmatrix(.j,(i-1)*scales(.j.)+k,l.):6:2, ' ');
                writeln;
              end; (*end k*) writeln;
              writeln('goahead : Press any key, correction : "%"');
              writeln; writeln;
              readln(ch);
              if ch = '%' then begin
                writeln('Enter Choice, row, col, prob  : ');
                readln(k, l, m, p);
                estmatrix(.j,(k-1)*scales(.j.)+l,m.) := p; end;
         until not(ch = '%');
        end; (*end j*)
end;

(* ========================================= *)
(* ===   Procedure Get State Matrix      === *)
(* ========================================= *)
```

```
procedure Getstatmatrix(numfactor:integer; possiblestates:integer;
                                scales:scaletable; scalevals:scalevaltable;
                        var statmatrix:statematrix);
var
     i, j, K,
     ptr, loop, mult  : integer;
begin
    mult := one;
    for i := one to numfactor do
     begin
        mult := mult * scales(.i.);
        loop := possiblestates div mult;
        ptr := 0;
        for j := one to mult do
          begin
            ptr := ptr + one;
            if ptr > scales(.i.) then ptr := (ptr mod scales(.i.));
            for k := one to loop do
              statmatrix(.(j-1)*loop+k.).column(.i.):=scalevals(.i,ptr.);
          end;
     end;
end;

(* ========================================= *)
(* ===   Procedure    Print State Matrix  === *)
(* ========================================= *)
procedure printstatmatrix(numfactor:integer; possiblestates:integer;
                                  statmatrix:statematrix);
var
     colptr,
     scaleptr, tableptr : integer;
begin
    page;
    writeln('______________________':57);
    writeln('Organizational  States':57);
    writeln('______________________':57);
    writeln; writeln;
    writeln('_______________________________________________':70);
    writeln('   Number of factors     : ':49, numfactor : 1);
    writeln('   State        Combinations of Scale Values':67);
    writeln('_______________________________________________':70);
    for tableptr := one to possiblestates do
      begin
       write(' ':23); write(tableptr:6); write(' ':9);
       for Colptr := one to numfactor do
         write( statmatrix(.tableptr.).column(.colptr.) : 6:2, ' ':2);
       writeln;
      end;
    writeln('_______________________________________________':70);
end;

(* ========================================= *)
(* ===   Procedure  State Matrix for Help === *)
(* ========================================= *)
procedure printhelpfile(numfactor:integer; possiblestates:integer;
                                  statmatrix:statematrix);
var
     colptr,
     scaleptr, tableptr : integer;
begin
    page;
    rewrite(helpfile, 'helpfile data');
    writeln(helpfile, 'Organizational States':38);
    writeln(helpfile, '_____________________':38);
    writeln(helpfile); writeln(helpfile);
    write  (helpfile, '____________________________________________');
```

```
    writeln(helpfile, '__________');
    for tableptr := one to possiblestates  do
      begin
       write(helpfile, tableptr:6);
       write(helpfile, ' ':9);
       for Colptr := one to numfactor do
        write(helpfile,statmatrix(.tableptr.).column(.colptr.):6:2);
       writeln(helpfile, ' ':2);
      end;
    write  (helpfile, '____________________________________________');
    writeln(helpfile, '__________');
end;


(* ========================================== *)
(* ==  Procedure  Transition Matrix        == *)
(* ========================================== *)
procedure gettranmatrix(numfactor:integer; possiblestates:integer;
                            choices:choicetable; scales:scaletable;
                            scalevals:scalevaltable;
                            estmatrix:estimationtable;
                            statmatrix:statematrix;
                        var tranmatrix:transitiontable);
var
    init, next,
    cho, columns,
    pointer, firstptr, secondptr : integer;
    getprob, multiprob : real;
    found : boolean;
begin
      for init := one to possiblestates  do   (* loop1 *)
        for cho  := one to choices(.init.)   do  (* loop2 *)
          for next := one to possiblestates     do (* loop3 *)
            begin
              multiprob:= 1;
              for columns := one to numfactor do             (* loop4 *)
               begin
                 pointer := 0;
                 firstptr := 0; secondptr := 0;
                 found := false;
                 While not found do
                  begin
                    pointer := pointer + 1;
                    if scalevals(.columns,pointer.) =
                        statmatrix(.init.).column(.columns.) then
                             firstptr := pointer;
                    if scalevals(.columns,pointer.) =
                       statmatrix(.next.).column(.columns.)  then
                             secondptr := pointer;
                    found := not ((firstptr*secondptr)=0)
                  end;
                secondptr := (cho-1)*scales(.columns.)+ secondptr;
                getprob   := estmatrix(.columns,secondptr,firstptr.);
                multiprob := multiprob * getprob;
               end;
            (* end of loop4 *)
            tranmatrix(.(init-1)*choices(.init.)+cho,next.) := multiprob;
            end;
          (* end of loop3 *)
        (* end of loop2 *)
      (* end of loop1 *)
end;


(* ========================================== *)
(* ==  PROCEDURE  PRINT  Transition Matrix == *)
(* ========================================== *)
```

```
procedure printtranmatrix(possiblestates:integer; choices:choicetable;
                              tranmatrix:transitiontable);
var
    columns,
    init, next : integer;
    temparray : array (.1..5.) of real;
begin
     for init := one to possiblestates  do (* loop1 *)
      begin
       page;
       writeln('_________________':47);
       writeln('Transition Matrix':47);
       writeln('_________________':47);
       writeln;
       writeln;
       writeln('Initial State : Z',init:3);
       write  ('________________________________________________________');
       writeln('_________________________');
       write  ('state        ');
       for columns := one to choices(.init.) do
         write('C(',columns:1,')',' ':10);
       writeln;
       write  ('________________________________________________________');
       writeln('_________________________');
         (* erase *)
         for columns := one to choices(.init.) do
           temparray(.columns.) := 0;
       for next  := one to possiblestates  do (* loop2 *)
        begin
         write(next:4);
         for columns := one to choices(.init.) do begin  (* loop3 *)
           write(tranmatrix(.(init-1)*choices(.init.)+columns, next.):
                                                                         14:6
           (* erase *)
           temparray(.columns.) := temparray(.columns.) +
                 tranmatrix(.(init-1)*choices(.init.)+columns, next.);
         end; (* erase *)
             (* end of loop3 *)
          writeln;
         end;
       (* end of loop2 *)
      write  ('--------------------------------------------------------');
      writeln('-----------------------');
      end;
     (* end of loop1 *)
end;


(* ======================================== *)
(* ===   Procedure Do Goodness Measure    === *)
(* ======================================== *)
procedure Goodnessmeasure(numfactor:integer; possiblestates:integer;
                              statmatrix:statematrix;
                          var goodmatrix:goodnessmatrix);
var
    index,
    ideal, anti  : integer;
    temp,
    max, min     : real;
(* ========================================= *)
(* ===   Procedure  Print Goodness Matrix  === *)
(* ========================================= *)
procedure printgoodmatrix;
const
    one = 1;
```

```
     two = 2;
     four= 4;
     eight=8;
var
     loop,
     row, col : integer;
begin
    page;
    writeln('                    __________________________           ');
    writeln('                    Goodness  Measurements               ');
    writeln('                    __________________________           ');
    writeln;
    writeln;
    write  ('____________________________________');
    writeln('________________________________');
    write  ('   State        Combination ');
    writeln(' ':(numfactor-two)*eight+four, 'Goodness Remarks');
    write  ('   Z(i)                ');
    for loop := one to numfactor do
        write('val',loop:1,' ':4);  writeln('value');
    write  ('____________________________________');
    writeln('________________________________');
    for row := one to possiblestates  do
       begin
        write(row:6);
        write(' ':9);
        for Col := one to numfactor do
           write( statmatrix(.row.).column(.col.) : 6:2, ' ':2);
        write(goodmatrix(.row.) : 6:2, ' ':2);
        if row = ideal then write(' ':5, 'ideal')
           else if row = anti then write(' ':5, 'anti-ideal');
        writeln;
       end;
    writeln('_________________________________________________________________');
end;

(** Procedure Begin **)
begin
    max := -999; min := 999;
    for index := one to possiblestates  do
      begin
        Case numfactor of
           2 : writeln('* Not prepared *');
           3 : begin
                 temp := -(statmatrix(.index.).column(.one.))
                             +(statmatrix(.index.).column(.two.))
                                +(statmatrix(.index.).column(.three.));
                 goodmatrix(.index.) := temp;
                 if temp > max then
                    begin
                      max := temp; ideal := index;
                    end;
                 if temp < min then
                    begin
                      min := temp; anti  := index;
                    end;
               end;
           4 : writeln('* Not prepared *');
        end; (* end case *)
     end; (* end for *)
     Printgoodmatrix;
end;


(* ========================================== *)
(* ==  Procedure  Get Benfit Matrix        == *)
(* ========================================== *)
procedure getbenematrix(possiblestates:integer;
```

```
                              choices:choicetable;
                              tranmatrix:transitiontable;
                              goodmatrix:goodnessmatrix;
                          var benematrix:benefittable);
var
     row, col, loop : integer;
     temp : real;
begin
      for row := one to possiblestates  do   (* loop1 *)
         for col := one to choices(.row.)    do      (* loop2 *)
           begin
              temp := 0;
              for loop  := one to possiblestates  do   (* loop3 *)
                temp := temp+(tranmatrix(.(row-1)*choices(.row.)+col,loop.)
                                *(goodmatrix(.loop.) - goodmatrix(.row.)));
              benematrix(.row,col.) := temp;
             (* end of loop3 *)
           end;
         (* end of loop2 *)
      (* end of loop1 *)
end;

(* ============================================= *)
(* ==  Procedure  Print  Benefit  Matrix   == *)
(* ============================================= *)
procedure printbenematrix(possiblestates:integer; maxcho:integer;
                                choices:choicetable; benematrix:benefittable);
var
     row, col : integer;
begin
   page;
   writeln('__________________':47);
   writeln('Benefit    Matrix':47);
   writeln('__________________':47);
   writeln;
   writeln;
   write  ('State Z(i)  ');
   for col := one to maxcho  do
     write('C(',col:1,')',' ':10);
   writeln;
   write  ('________________________________________________________');
   writeln('________________________');
   for row := one to possiblestates  do (* loop1 *)
     begin
       write(row:4);
       for col := one to choices(.row.)  do (* loop2 *)
           write(benematrix(.row,col.):14:6);
       (* end of loop2 *)
       writeln;
     end;
   (* end of loop1 *)
   write  ('________________________________________________________');
   writeln('________________________');
end;

(***  PART II.  MAIN PROGRAM  ***)
begin
    (* ======== Input Data via Terminal ======= *)
    (* Step I.   *)
    Getnumoffactor (numfactor,possiblestates,scales);
    Getnumofchoices(numfactor,possiblestates,choices,maxcho);
    Getestprob     (numfactor,maxcho,scales,scalevals,estmatrix);

    (* ======== Get Transition Matrix ========= *)
    (* Step II.  *)
    Getstatmatrix(numfactor,possiblestates,scales,scalevals,statmatrix);
    printstatmatrix(numfactor,possiblestates,statmatrix);
```

```
    printhelpfile  (numfactor,possiblestates,statmatrix);
    (* Step III. *)
    Gettranmatrix(numfactor,possiblestates,choices,scales,scalevals,
                     estmatrix,statmatrix,tranmatrix);
    Printtranmatrix(possiblestates,choices,tranmatrix);

    (* ======== Get Benefit    Matrix ========= *)
    (* Step  IV. *)
    Goodnessmeasure(numfactor,possiblestates,statmatrix,goodmatrix);

    (* Step  V.  *)
    Getbenematrix  (possiblestates,choices,tranmatrix,
                       goodmatrix, benematrix);
    printbenematrix(possiblestates,maxcho,choices,benematrix);
end;
```

```
(* ##########**##################################### *)
(* ###                                          ### *)
(* ###     G  C  M           PART    III.       ### *)
(* ###        Generate Long Run Policy          ### *)
(* ### This part processes howard's algorithm ### *)
(* ### main modules are Dynamic formulation,  ### *)
(* ### Resolve variable, and Setnewaction     ### *)
(* ################################################ *)

(*******************************************)
(*****   GET    CHOICE     POLICY      *****)
(*******************************************)
Procedure Getchoice(possiblestates:integer; choices:choicetable;
                        tranmatrix:transitiontable;
                        benematrix:benefittable; var policy:policytable);
type
    markovtable = array (.one..maxstate, one..maxstateetc.) of real;
    determinval = array (.one..maxstateetc.) of real;
var
    index               : integer;
    i,j                 : integer;
    maxincome           : real;
    markov              : markovtable;
    resolution          : determinval;
    newpolicy           : policytable;
    reached, matched,
    quitnow, inforced : boolean;
(* ============================================ *)
(* ==  PROCEDURE  Initialize  policy Table  == *)
(* ============================================ *)
procedure initialize(possiblestates:integer; choices:choicetable;
                         benematrix:benefittable; var policy:policytable);
var
    row, col : integer;
    max : real;
begin
    for row := one to possiblestates  do
      begin
        max := -999;
        for col := one to choices(.row.) do
         If benematrix(.row,col.) > max then
           begin
            max := benematrix(.row,col.);
            policy(.row.) := col;
           end;
      end;
end;
(* ============================================ *)
(* ==  PROCEDURE  Dynamic Formulation       == *)
(* ============================================ *)
procedure dynamic_formulation(vi:integer; possiblestates:integer;
                              choices:choicetable; tranmatrix:transitiontable;
                              benematrix:benefittable; policy:policytable;
                         var markov:markovtable);
var
    row, col : integer;
begin
  (* fill matrix with variable coefficient *)
  for row := one to possiblestates do
      markov(.row,one.) := 1.0;
  for row := one to possiblestates do
     for col := one to possiblestates  do
       If row = col then
          markov(.row,col+1.) :=
             1-(tranmatrix(.(row-1)*choices(.row.)+policy(.row.),col.))
       else
```

```
            markov(.row,col+1.) :=
                -(tranmatrix(.(row-1)*choices(.row.)+policy(.row.),col.));
   for row := one to possiblestates do
      markov(.row,possiblestates+2.) := benematrix(.row,policy(.row.).);
   (* remove an arbitary vatiable v(i) *)
   for row := one to possiblestates do
      for col := vi + one to possiblestates + one do
         markov(.row,col.) := markov(.row,col+1.);
end;

(* ============================================ *)
(* ==  PROCEDURE  Resolve all  variables    == *)
(* ============================================ *)
procedure Resolvevariable(vi:integer; n:integer;
                          var markov:markovtable;
                          var resolution:determinval);
var
    i,j,k      : integer;
    multfac,
    temp       : real;
begin
   (********* manipulate markov matrix ***********)
   for i := 1 to n-1 do begin
     if markov(.i,i.) <> 1 then
       begin
         multfac := 1/(markov(.i,i.));
         for j := i to n+1 do
           markov(.i,j.) := markov(.i,j.) * multfac;
       end;
     for j := i+1 to n do
       begin
         multfac := markov(.j,i.);
         for k := i to n+1 do
           markov(.j,k.) := markov(.j,k.) - (multfac * markov(.i,k.));
       end;
   end;
   (********* find solution ************)
   resolution(.n.) := markov(.n,n+1.)/markov(.n,n.);
   for i := n-1 downto 1 do
     begin
       temp := 0;
       for j := i+1 to n do
         temp := temp +
                      (markov(.i,j.) * resolution(.j.));
       resolution(.i.) := markov(.i,n+1.) - temp;
     end;
   (*******************************************)
   for j := n downto vi + 1 do
     resolution(.j+1.) := resolution(.j.);
   resolution(.vi+1.) := 0;
end;

(* ============================================ *)
(* ==  PROCEDURE  Set New Action           == *)
(* ============================================ *)
procedure setnewaction(possiblestates:integer; choices:choicetable;
                       tranmatrix:transitiontable;
                       benematrix:benefittable;
                       resolution:determinval; var newpolicy:policytable;
                       policy:policytable);
type
   eval_property = record
      ptr : integer;
      val : real;
   end;
var
```

```
    col, cho, loop,
    i,j,k,selectcode: integer;
    temp,dynamic,max: real;
    property        : array (.one..maxchoice.) of eval_property;
begin
  for loop := one to possiblestates  do
    begin
      max := -999;
      for cho := one to choices(.loop.)  do
        begin
          dynamic := 0;
          for col := one to possiblestates      do begin
            dynamic:=dynamic +
              (tranmatrix(.(loop-1)*choices(.loop.)+cho,col.) *
                                                  resolution(.col+one.));
          end;
          dynamic:=dynamic + benematrix(.loop,cho.) ;
          property(.cho.).ptr := cho;
          property(.cho.).val := dynamic;
        end; (* end of choice *)
      (* sorting *)
      for i := choices(.loop.) downto two do
       for j := one to i-1 do
        if property(.i.).val > property(.j.).val then begin
           temp := property(.j.).val;
           property(.j.).val := property(.i.).val;
           property(.i.).val := temp;
           k     := property(.j.).ptr;
           property(.j.).ptr := property(.i.).ptr;
           property(.i.).ptr := k; end;
     (** equality **)
       if (property(.1.).val >= 0) and (property(.2.).val >= 0) then
          selectcode := 1;
       if (property(.1.).val >= 0) and (property(.2.).val <  0) then
          selectcode := 2;
       if (property(.1.).val < 0) then
          selectcode := 3;
       case selectcode of
         1 : if ((property(.1.).val-property(.2.).val)/
                 property(.1.).val) <= 0.0001 then  (* identity *)
                 newpolicy(.loop.) := policy(.loop.)
             else newpolicy(.loop.) := property(.1.).ptr;
         2 : newpolicy(.loop.) := property(.1.).ptr;(* nonidentity *)
         3 : if ((property(.2.).val-property(.1.).val)/
                 property(.1.).val) > 0.0001 then  (* nonidentity *)
                 newpolicy(.loop.) := property(.1.).ptr
             else newpolicy(.loop.) := policy(.loop.)
       end;
    end; (* end of loop *)
end;

(*** PART III    MAIN PROGRAM  ***)
begin
  vi := possiblestates;
  repeat
    (* Step I.   *)
    Initialize(possiblestates,choices,benematrix,policy);
    maxincome := -999.0;
    inforced := false;
    repeat
      (* Step II.  *)
      Dynamic_formulation(vi,possiblestates,choices,
                          tranmatrix, benematrix, policy, markov);
      (* Step III. *)
      Resolvevariable(vi,possiblestates,markov,resolution);
      for i := 1 to possiblestates + 1 do
      (* Step IV.  *)
      setnewaction(possiblestates,choices,tranmatrix,benematrix,
```

```
                    resolution,newpolicy,policy);
      quitnow  := false;
      if maxincome <= resolution(.one.) then
        begin
          index := 0;
          quitnow := true;
          repeat
            index := index + 1;
            reached := (index >= possiblestates);
            matched := (policy(.index.)=newpolicy(.index.));
            quitnow := quitnow and matched;
            policy(.index.) := newpolicy(.index.);
          until reached;
          if quitnow and (resolution(.one.) < 0) then inforced := true
            else maxincome := resolution(.one.);
        end
      else inforced := true; (*endif maxincome*)
  until (quitnow or inforced);
  vi := vi-1;
  if (vi = 0) and (not quitnow) then writeuser(IMPOSSIBLE);
 until ((quitnow and (not inforced)) or (vi=0));
 vi := vi + 1;
 gain := resolution(.one.); (*printed on policytable*)
end;
```

```
(* ########################################### *)
(* ###                                     ### *)
(* ###   G  C  M           PART    IV.     ### *)
(* ###                                     ### *)
(* ########################################### *

(********************************************)
(****      I N T E R A C T I O N         ***)
(********************************************)
Procedure interaction(possiblestates:integer; policy:policytable);
type
    elementtable  = array(.one..maxfactor.) of real;
    oneline       = packed array(.one..seventy.) of char;
var
    current   : integer;
    elements : elementtable;
    line      : oneline;
    yesno     : char;
    doall, nomore,
    needhelp : boolean;

(* ========================================== *)
(* ==    Do  Partial Decision policy       == *)
(* ========================================== *)
procedure dopartial(possiblestates:integer; policy:policytable);
begin
  writeln('Do you want to see Helpfile for current problem ? y/n');
  readln(yesno);
  needhelp := (yesno='Y') or (yesno='y');
  if needhelp then begin
      reset (helpfile, 'helpfile data');
      while not eof(helpfile) do begin
        readln(helpfile, line);
        writeln(line); end;
      end;
  repeat
    writeln('What is your current state ?  ===> Type n');
    readln(current);
  until (current > 0) and (current <= possiblestates);
  writeln;
  write  ('Current State : ', current:2, ',    '  );
  writeln('Best   Choice : ', policy(.current.):2);
  writeln;
end;
(* ========================================== *)
(* ==    PRINT  POLICY  TABLE              == *)
(* ========================================== *)
procedure printpolicy(possiblestates:integer; policy:policytable);
var
    loop, cnt : integer;
begin
  page;
  writeln('_________________':27);
  writeln('Decision  Choices':27);
  writeln('_________________':27);
  writeln;
  writeln;
  writeln('State                                  Choice Policy');
  writeln('Z(i)                                       C(k)     ');
  writeln('____________________________________________________');
  writeln;
  cnt := 0;
  for loop := one to possiblestates  do    (* loop2 *)
    begin
      writeln(loop:4, ' ':30, policy(.loop.):3);
```

```
        cnt := cnt + 1;
        if (cnt mod 3) = 0 then readln(wait);
      end;
    writeln('___________________________________________');
    writeln;
    writeln('set arbitary v(', vi:2, ')  to 0');
    writeln;
    writeln('maxincome per period : ', gain);
    writeln;
    page;
  end;

  begin
    repeat
      page;
      writeln('Do you want all decision policy ? y/n');
      readln(yesno);
      doall  := (yesno='Y') or (yesno='y');

      if doall then printpolicy(possiblestates,policy)
        else dopartial(possiblestates,policy);

      writeln('Do you have another ploblem ?');
      readln(yesno);
      nomore := (yesno='N') or (yesno='n');
    until nomore;
  end;
```

```
(* ############################################ *)
(* ###                                     ### *)
(* ###   G  C  M            PART     V.    ### *)
(* ###        Exit Garbage Can Prog.       ### *)
(* ############################################ *

(*********************************************)
(****      GET     BOOLEAN    VALUE       ***)
(*********************************************)
Procedure Getboolean(var Exit:boolean);
begin
  Exit := true;
  page;
end;


(*********************************************)
(***  End  of Garbage  Can  Model Program   ***)
(*********************************************)




(*********************************************)
(*****   M A I N         P R O G R A M   *****)
(*********************************************)
begin
  Quitnow := false;
  While not quitnow do begin
    if getcommand(command) then
     Case command of
       EXECGCM  : begin
                    Getmatrix(possiblestates,choices,
                                    tranmatrix,benematrix);
                    Getchoice(possiblestates,choices,
                                tranmatrix,benematrix,policy);
                    Interaction(possiblestates,policy);
                  end;
       EXECEXIT : Getboolean(Quitnow);
     end
    else Writeuser(BADLINE);
  end;
end.
```

APPENDIX B
USER MANUAL

```
PGCM Program, 1987

I.  PROGRAM NAME : pgcmprog (written in Waterloo pascal language)
II. PURPOSE : Get a set of choices available in a specific organization
              problem
III.TO USE:

     1. Before Execution
         1) Formulate problem and set alternative actions
         2) Turn on your terminal
         3) LOGIN userid
         4) ENTER PASSWORD(IT WILL NOT APPEAR WHEN TYPED):
         5) Memory extention (if necessary)
            : DEF STOR 1500k
            : I CMS
         6) Execute garbage can program
            : pw gcmprog pascal
     2. During Execution
         1) Select menu option
            1 ------ ExecGCM
            2 ------ ExitGCM
         2) Enter number of scale points for each factor
            i.e> factor 1  : 3
                 factor 2  : 2
                 factor 3  : 2
            ======> possible states  : 3 x 2 x 2 = 12
         4) Enter number of choices for each state
            i.e> state 1 ? 2
                 state 2 ? 2
                 state 3 ? 3
                 state 4 ? 3
                 state 5 ? 3
                 state 6 ? 3
                 state 7 ? 2
                 state 8 ? 2
                 state 9 ? 3
                 state 10? 3
                 state 11? 3
                 state 12? 3
            ======> maxchoices  : 3
         4) Enter estimation probabilities
            if user has the following input data,
```

factor1(P)

cho	P 0.0	0.5	1.0
1	0.2	0.4	0.3
1	0.3	0.5	0.5
1	0.5	0.1	0.2

factor2(S)

cho	S 0.0	1.0
1	0.1	0.6
1	0.9	0.4

factor3(E)

cho	E 0.0	1.0
1	0.3	0.6
1	0.7	0.4

cho	0.0	0.5	1.0
2	0.1	0.3	0.2
2	0.5	0.3	0.2
2	0.4	0.4	0.6

cho	0.0	1.0
2	0.5	0.1
2	0.5	0.9

cho	0.0	1.0
2	0.2	0.9
2	0.8	0.1

cho	0.0	0.5	1.0
3	0.1	0.7	0.2

cho	0.0	1.0
3	0.8	0.3

cho	0.0	1.0
3	0.4	0.5

```
3 | 0.1 0.2 0.3      3  | 0.2 0.7       3 | 0.6 0.5
3 | 0.8 0.1 0.5
```

Interact with the folowing manner

```
(factor1)
  Initial  State f(1) =  0.00  Choice C(1)
  Terminal State
   prob1   prob2   prob3 ?
   ________________________
   |0.2     0.3     0.5   |
   |______________________|
  * rectangle : input data
  Initial  State f(1) =  0.50  Choice C(1)
  Terminal State
   prob1   prob2   prob3 ?
   ________________________
   |0.4     0.5     0.1   |
   |______________________|
  Initial  State f(1) =  0.10  Choice C(1)
  Terminal State
   prob1   prob2   prob3 ?
   ________________________
   |0.3     0.5     0.2   |
   |______________________|

(factor2)
    ......................
    ......................
    ......................
(factor3)
    ......................
    ......................
    ......................
* choice 2 and choice 3 operations are the same
```

3. After Execution

```
  1) Do you want all decision policy ? y/n
     yes
     ___
          display all choice policies
     no
     ___
          Do you want to see Helpfile for current problem ? y/n
          yes
          ___
             display  organizational states for memory aids
             What is your current state ?  ===> Type n
          no
          ___
             What is your current state ?  ===> Type n

  2) Do you have another ploblem ?
     yes
     ___
       go to procedure 3.1
     no
     ___
       go to procedure 2.1
```

4. Get results

```
  1) PRINT OUT
     : PRINT gcmprog listing
```

2) BROWSE
: FLIST
: use PF key

5. Turn off
: LOGOFF

APPENDIX C
OFFENSIVE OPERATION EXAMPLE

```
**********************************
**                              **
** Step I  : User   Interaction **
**                              **
**********************************
```

I. Formulate problem and prepare estimation probabilities

P : Importance of problems to be solved
S : Degree of effectiveness in problem-solving
E : Potential energy of participants

p1=0 No significant problem regarding attack forces, mission load, weather, relation with consequent military operation

p2=.5 Moderate shortage of attacking forces, not good weapon system good weather, a certain time delay to the consequent operation

p3=1 Acute shortage of attacking forces, big mission load, bad weather, a tremendous time delay to the consequent operation

S1=0 Most of personnel have no experience in the battle field, poor coordination with adjacent unit, poor performance weapon systems

S2=.5 Some personnel have an experience in the battle field, appropriate coordination and reasonable attack-defense forces ratio, good performance weapon systems, good logistic support systems

S3=1 Some personnel have an experience in the battle field, excellent coordination, best attack-defense forces ratio, excellent performance weapon systems , sufficient logistic support systems

E1=0 Not quite proud of their operations,passive action

E2=1 High morale, high responsibility

	P			S			E	
Choice	0.00	0.50	1.00	0.00	0.50	1.00	0.00	1.00
1	0.98	0.19	0.01	0.97	0.01	0.01	0.95	0.01
1	0.01	0.80	0.02	0.02	0.70	0.01	0.05	0.99
1	0.01	0.01	0.97	0.01	0.29	0.98		
2	0.98	0.70	0.10	0.20	0.01	0.01	0.20	0.01
2	0.01	0.29	0.50	0.70	0.19	0.01	0.80	0.99
2	0.01	0.01	0.40	0.10	0.80	0.98		
3	0.98	0.80	0.39	0.90	0.01	0.01	0.70	0.05
3	0.01	0.19	0.60	0.09	0.39	0.01	0.30	0.95

3	0.01	0.01	0.01	0.01	0.60	0.98		
4	0.30	0.10	0.20	0.30	0.20	0.30	0.20	0.30
4	0.60	0.80	0.60	0.60	0.60	0.40	0.80	0.70
4	0.10	0.10	0.20	0.10	0.20	0.30		

II. Execution

```
Select Menu Option
______________________
1. ExecGCM (Execute GCM Program)
2. ExecExit(Execution stop     )

Enter # of scale points for each factor :
_________________________________________
factor 1 :              3
factor 2 :              3
factor 3 :              2
goahead : Press any key, correction : "%"

Get Number of Choices for each State Z(i)   i = 1 .. 18
________________________________________________________
State Z(i)         # of choices
__________         ____________
       1 ?                4
       2 ?                4
       3 ?                4
       4 ?                4
       5 ?                4
       6 ?                4
       7 ?                4
       8 ?                4
       9 ?                4
      10 ?                4
      11 ?                4
      12 ?                4
      13 ?                4
      14 ?                4
      15 ?                4
      16 ?                4
      17 ?                4
      18 ?                4
goahead : Press any key, correction : "%"

Get Estimation Probabilities
____________________________
(factor1)
Initial  State f(1) =   0.00  Choice  C(1)
Terminal State :
 prob1  prob2  prob3 ?  0.98 0.19 0.01
Initial  State f(1) =   0.50  Choice  C(1)
Terminal State :
 prob1  prob2  prob3 ?  0.01 0.80 0.02
Initial  State f(1) =   1.00  Choice  C(1)
Terminal State :
 prob1  prob2  prob3 ?  0.01 0.01 0.97
```

```
          (* Display Input Data *)
              ______________________
                0.00    0.50    1.00
              ______________________
     1        0.98    0.19    0.01
     1        0.01    0.80    0.02
     1        0.01    0.01    0.97
goahead : Press any key, correction : "%"
 .......................................
 .......................................
 Repeat this step for remaining (n-1)factors
 .......................................
 .......................................
```

```
(**********************************)
(**                              **)
(** Step II : Generate  Matrices **)
(**                              **)
(**********************************)
```

Transition Matrix

Initial State : Z 1

state	C(1)	C(2)	C(3)	C(4)
1	0.903070	0.039200	0.617400	0.018000
2	0.047530	0.156800	0.264600	0.072000
3	0.018620	0.137200	0.061740	0.036000
4	0.000980	0.548800	0.026460	0.144000
5	0.009310	0.019600	0.006860	0.006000
6	0.000490	0.078400	0.002940	0.024000
7	0.009215	0.000400	0.006300	0.036000
8	0.000485	0.001600	0.002700	0.144000
9	0.000190	0.001400	0.000630	0.072000
10	0.000010	0.005600	0.000270	0.288000
11	0.000095	0.000200	0.000070	0.012000
12	0.000005	0.000800	0.000030	0.048000
13	0.009215	0.000400	0.006300	0.006000
14	0.000485	0.001600	0.002700	0.024000
15	0.000190	0.001400	0.000630	0.012000
16	0.000010	0.005600	0.000270	0.048000
17	0.000095	0.000200	0.000070	0.002000
18	0.000005	0.000800	0.000030	0.008000

Initial State : Z 2

state	C(1)	C(2)	C(3)	C(4)
1	0.009506	0.001960	0.044100	0.027000
2	0.941094	0.194040	0.837900	0.063000
3	0.000196	0.006860	0.004410	0.054000
4	0.019404	0.679140	0.083790	0.126000
5	0.000098	0.000980	0.000490	0.009000
6	0.009702	0.097020	0.009310	0.021000
7	0.000097	0.000020	0.000450	0.054000
8	0.009603	0.001980	0.008550	0.126000
9	0.000002	0.000070	0.000045	0.108000
10	0.000198	0.006930	0.000855	0.252000
11	0.000001	0.000010	0.000005	0.018000
12	0.000099	0.000990	0.000095	0.042000
13	0.000097	0.000020	0.000450	0.009000
14	0.009603	0.001980	0.008550	0.021000
15	0.000002	0.000070	0.000045	0.018000
16	0.000198	0.006930	0.000855	0.042000
17	0.000001	0.000010	0.000005	0.003000
18	0.000099	0.000990	0.000095	0.007000

Initial State : Z 3

state	C(1)	C(2)	C(3)	C(4)
1	0.009310	0.001960	0.006860	0.012000
2	0.000490	0.007840	0.002940	0.048000
3	0.651700	0.037240	0.267540	0.036000
4	0.034300	0.148960	0.114660	0.144000
5	0.269990	0.156800	0.411600	0.012000
6	0.014210	0.627200	0.176400	0.048000
7	0.000095	0.000020	0.000070	0.024000
8	0.000005	0.000080	0.000030	0.096000
9	0.006650	0.000380	0.002730	0.072000
10	0.000350	0.001520	0.001170	0.288000
11	0.002755	0.001600	0.004200	0.024000
12	0.000145	0.006400	0.001800	0.096000
13	0.000095	0.000020	0.000070	0.004000
14	0.000005	0.000080	0.000030	0.016000
15	0.006650	0.000380	0.002730	0.012000
16	0.000350	0.001520	0.001170	0.048000
17	0.002755	0.001600	0.004200	0.004000
18	0.000145	0.006400	0.001800	0.016000

Initial State : Z 4

state	C(1)	C(2)	C(3)	C(4)
1	0.000098	0.000098	0.000490	0.018000
2	0.009702	0.009702	0.009310	0.042000
3	0.006860	0.001862	0.019110	0.054000
4	0.679140	0.184338	0.363090	0.126000
5	0.002842	0.007840	0.029400	0.018000
6	0.281358	0.776160	0.558600	0.042000
7	0.000001	0.000001	0.000005	0.036000
8	0.000099	0.000099	0.000095	0.084000
9	0.000070	0.000019	0.000195	0.108000
10	0.006930	0.001881	0.003705	0.252000
11	0.000029	0.000080	0.000300	0.036000
12	0.002871	0.007920	0.005700	0.084000
13	0.000001	0.000001	0.000005	0.006000
14	0.000099	0.000099	0.000095	0.014000
15	0.000070	0.000019	0.000195	0.018000
16	0.006930	0.001881	0.003705	0.042000
17	0.000029	0.000080	0.000300	0.006000
18	0.002871	0.007920	0.005700	0.014000

Initial State : Z 5

state	C(1)	C(2)	C(3)	C(4)
1	0.009310	0.001960	0.006860	0.018000
2	0.000490	0.007840	0.002940	0.072000
3	0.009310	0.001960	0.006860	0.024000
4	0.000490	0.007840	0.002940	0.096000
5	0.912380	0.192080	0.672280	0.018000
6	0.048020	0.768320	0.288120	0.072000
7	0.000095	0.000020	0.000070	0.036000
8	0.000005	0.000080	0.000030	0.144000
9	0.000095	0.000020	0.000070	0.048000
10	0.000005	0.000080	0.000030	0.192000
11	0.009310	0.001960	0.006860	0.036000
12	0.000490	0.007840	0.002940	0.144000
13	0.000095	0.000020	0.000070	0.006000
14	0.000005	0.000080	0.000030	0.024000
15	0.000095	0.000020	0.000070	0.008000
16	0.000005	0.000080	0.000030	0.032000
17	0.009310	0.001960	0.006860	0.006000
18	0.000490	0.007840	0.002940	0.024000

Initial State : Z 6

state	C(1)	C(2)	C(3)	C(4)
1	0.000098	0.000098	0.000490	0.027000
2	0.009702	0.009702	0.009310	0.063000
3	0.000098	0.000098	0.000490	0.036000
4	0.009702	0.009702	0.009310	0.084000
5	0.009604	0.009604	0.048020	0.027000
6	0.950796	0.950796	0.912380	0.063000
7	0.000001	0.000001	0.000005	0.054000
8	0.000099	0.000099	0.000095	0.126000
9	0.000001	0.000001	0.000005	0.072000
10	0.000099	0.000099	0.000095	0.168000
11	0.000098	0.000098	0.000490	0.054000
12	0.009702	0.009702	0.009310	0.126000
13	0.000001	0.000001	0.000005	0.009000
14	0.000099	0.000099	0.000095	0.021000
15	0.000001	0.000001	0.000005	0.012000
16	0.000099	0.000099	0.000095	0.028000
17	0.000098	0.000098	0.000490	0.009000
18	0.009702	0.009702	0.009310	0.021000

Initial State : Z 7

state	C(1)	C(2)	C(3)	C(4)
1	0.175085	0.028000	0.504000	0.006000
2	0.009215	0.112000	0.216000	0.024000
3	0.003610	0.098000	0.050400	0.012000
4	0.000190	0.392000	0.021600	0.048000
5	0.001805	0.014000	0.005600	0.002000
6	0.000095	0.056000	0.002400	0.008000
7	0.737200	0.011600	0.119700	0.048000
8	0.038800	0.046400	0.051300	0.192000
9	0.015200	0.040600	0.011970	0.096000
10	0.000800	0.162400	0.005130	0.384000
11	0.007600	0.005800	0.001330	0.016000
12	0.000400	0.023200	0.000570	0.064000
13	0.009215	0.000400	0.006300	0.006000
14	0.000485	0.001600	0.002700	0.024000
15	0.000190	0.001400	0.000630	0.012000
16	0.000010	0.005600	0.000270	0.048000
17	0.000095	0.000200	0.000070	0.002000
18	0.000005	0.000800	0.000030	0.008000

Initial State : Z 8

state	C(1)	C(2)	C(3)	C(4)
1	0.001843	0.001400	0.036000	0.009000
2	0.182457	0.138600	0.684000	0.021000
3	0.000038	0.004900	0.003600	0.018000
4	0.003762	0.485100	0.068400	0.042000
5	0.000019	0.000700	0.000400	0.003000
6	0.001881	0.069300	0.007600	0.007000
7	0.007760	0.000580	0.008550	0.072000
8	0.768240	0.057420	0.162450	0.168000
9	0.000160	0.002030	0.000855	0.144000
10	0.015840	0.200970	0.016245	0.336000
11	0.000080	0.000290	0.000095	0.024000
12	0.007920	0.028710	0.001805	0.056000
13	0.000097	0.000020	0.000450	0.009000
14	0.009603	0.001980	0.008550	0.021000
15	0.000002	0.000070	0.000045	0.018000
16	0.000198	0.006930	0.000855	0.042000
17	0.000001	0.000010	0.000005	0.003000
18	0.000099	0.000990	0.000095	0.007000

Initial State : Z 9

state	C(1)	C(2)	C(3)	C(4)
1	0.001805	0.001400	0.005600	0.004000
2	0.000095	0.005600	0.002400	0.016000
3	0.126350	0.026600	0.218400	0.012000
4	0.006650	0.106400	0.093600	0.048000
5	0.052345	0.112000	0.336000	0.004000
6	0.002755	0.448000	0.144000	0.016000
7	0.007600	0.000580	0.001330	0.032000
8	0.000400	0.002320	0.000570	0.128000
9	0.532000	0.011020	0.051870	0.096000
10	0.028000	0.044080	0.022230	0.384000
11	0.220400	0.046400	0.079800	0.032000
12	0.011600	0.185600	0.034200	0.128000
13	0.000095	0.000020	0.000070	0.004000
14	0.000005	0.000080	0.000030	0.016000
15	0.006650	0.000380	0.002730	0.012000
16	0.000350	0.001520	0.001170	0.048000
17	0.002755	0.001600	0.004200	0.004000
18	0.000145	0.006400	0.001800	0.016000

Initial State : Z 10

state	C(1)	C(2)	C(3)	C(4)
1	0.000019	0.000070	0.000400	0.006000
2	0.001881	0.006930	0.007600	0.014000
3	0.001330	0.001330	0.015600	0.018000
4	0.131670	0.131670	0.296400	0.042000
5	0.000551	0.005600	0.024000	0.006000
6	0.054549	0.554400	0.456000	0.014000
7	0.000080	0.000029	0.000095	0.048000
8	0.007920	0.002871	0.001805	0.112000
9	0.005600	0.000551	0.003705	0.144000
10	0.554400	0.054549	0.070395	0.336000
11	0.002320	0.002320	0.005700	0.048000
12	0.229680	0.229680	0.108300	0.112000
13	0.000001	0.000001	0.000005	0.006000
14	0.000099	0.000099	0.000095	0.014000
15	0.000070	0.000019	0.000195	0.018000
16	0.006930	0.001881	0.003705	0.042000
17	0.000029	0.000080	0.000300	0.006000
18	0.002871	0.007920	0.005700	0.014000

Initial State : Z 11

state	C(1)	C(2)	C(3)	C(4)
1	0.001805	0.001400	0.005600	0.006000
2	0.000095	0.005600	0.002400	0.024000
3	0.001805	0.001400	0.005600	0.008000
4	0.000095	0.005600	0.002400	0.032000
5	0.176890	0.137200	0.548800	0.006000
6	0.009310	0.548800	0.235200	0.024000
7	0.007600	0.000580	0.001330	0.048000
8	0.000400	0.002320	0.000570	0.192000
9	0.007600	0.000580	0.001330	0.064000
10	0.000400	0.002320	0.000570	0.256000
11	0.744800	0.056840	0.130340	0.048000
12	0.039200	0.227360	0.055860	0.192000
13	0.000095	0.000020	0.000070	0.006000
14	0.000005	0.000080	0.000030	0.024000
15	0.000095	0.000020	0.000070	0.008000
16	0.000005	0.000080	0.000030	0.032000
17	0.009310	0.001960	0.006860	0.006000
18	0.000490	0.007840	0.002940	0.024000

Initial State : Z 12

state	C(1)	C(2)	C(3)	C(4)
1	0.000019	0.000070	0.000400	0.009000
2	0.001881	0.006930	0.007600	0.021000
3	0.000019	0.000070	0.000400	0.012000
4	0.001881	0.006930	0.007600	0.028000
5	0.001862	0.006860	0.039200	0.009000
6	0.184338	0.679140	0.744800	0.021000
7	0.000080	0.000029	0.000095	0.072000
8	0.007920	0.002871	0.001805	0.168000
9	0.000080	0.000029	0.000095	0.096000
10	0.007920	0.002871	0.001805	0.224000
11	0.007840	0.002842	0.009310	0.072000
12	0.776160	0.281358	0.176890	0.168000
13	0.000001	0.000001	0.000005	0.009000
14	0.000099	0.000099	0.000095	0.021000
15	0.000001	0.000001	0.000005	0.012000
16	0.000099	0.000099	0.000095	0.028000
17	0.000098	0.000098	0.000490	0.009000
18	0.009702	0.009702	0.009310	0.021000

Initial State : Z 13

state	C(1)	C(2)	C(3)	C(4)
1	0.009215	0.004000	0.245700	0.012000
2	0.000485	0.016000	0.105300	0.048000
3	0.000190	0.014000	0.024570	0.024000
4	0.000010	0.056000	0.010530	0.096000
5	0.000095	0.002000	0.002730	0.004000
6	0.000005	0.008000	0.001170	0.016000
7	0.018430	0.020000	0.378000	0.036000
8	0.000970	0.080000	0.162000	0.144000
9	0.000380	0.070000	0.037800	0.072000
10	0.000020	0.280000	0.016200	0.288000
11	0.000190	0.010000	0.004200	0.012000
12	0.000010	0.040000	0.001800	0.048000
13	0.893855	0.016000	0.006300	0.012000
14	0.047045	0.064000	0.002700	0.048000
15	0.018430	0.056000	0.000630	0.024000
16	0.000970	0.224000	0.000270	0.096000
17	0.009215	0.008000	0.000070	0.004000
18	0.000485	0.032000	0.000030	0.016000

Initial State : Z 14

state	C(1)	C(2)	C(3)	C(4)
1	0.000097	0.000200	0.017550	0.018000
2	0.009603	0.019800	0.333450	0.042000
3	0.000002	0.000700	0.001755	0.036000
4	0.000198	0.069300	0.033345	0.084000
5	0.000001	0.000100	0.000195	0.006000
6	0.000099	0.009900	0.003705	0.014000
7	0.000194	0.001000	0.027000	0.054000
8	0.019206	0.099000	0.513000	0.126000
9	0.000004	0.003500	0.002700	0.108000
10	0.000396	0.346500	0.051300	0.252000
11	0.000002	0.000500	0.000300	0.018000
12	0.000198	0.049500	0.005700	0.042000
13	0.009409	0.000800	0.000450	0.018000
14	0.931491	0.079200	0.008550	0.042000
15	0.000194	0.002800	0.000045	0.036000
16	0.019206	0.277200	0.000855	0.084000
17	0.000097	0.000400	0.000005	0.006000
18	0.009603	0.039600	0.000095	0.014000

Initial State : Z 15

state	C(1)	C(2)	C(3)	C(4)
1	0.000095	0.000200	0.002730	0.008000
2	0.000005	0.000800	0.001170	0.032000
3	0.006650	0.003800	0.106470	0.024000
4	0.000350	0.015200	0.045630	0.096000
5	0.002755	0.016000	0.163800	0.008000
6	0.000145	0.064000	0.070200	0.032000
7	0.000190	0.001000	0.004200	0.024000
8	0.000010	0.004000	0.001800	0.096000
9	0.013300	0.019000	0.163800	0.072000
10	0.000700	0.076000	0.070200	0.288000
11	0.005510	0.080000	0.252000	0.024000
12	0.000290	0.320000	0.108000	0.096000
13	0.009215	0.000800	0.000070	0.008000
14	0.000485	0.003200	0.000030	0.032000
15	0.645050	0.015200	0.002730	0.024000
16	0.033950	0.060800	0.001170	0.096000
17	0.267235	0.064000	0.004200	0.008000
18	0.014065	0.256000	0.001800	0.032000

Initial State : Z 16

state	C(1)	C(2)	C(3)	C(4)
1	0.000001	0.000010	0.000195	0.012000
2	0.000099	0.000990	0.003705	0.028000
3	0.000070	0.000190	0.007605	0.036000
4	0.006930	0.018810	0.144495	0.084000
5	0.000029	0.000800	0.011700	0.012000
6	0.002871	0.079200	0.222300	0.028000
7	0.000002	0.000050	0.000300	0.036000
8	0.000198	0.004950	0.005700	0.084000
9	0.000140	0.000950	0.011700	0.108000
10	0.013860	0.094050	0.222300	0.252000
11	0.000058	0.004000	0.018000	0.036000
12	0.005742	0.396000	0.342000	0.084000
13	0.000097	0.000040	0.000005	0.012000
14	0.009603	0.003960	0.000095	0.028000
15	0.006790	0.000760	0.000195	0.036000
16	0.672210	0.075240	0.003705	0.084000
17	0.002813	0.003200	0.000300	0.012000
18	0.278487	0.316800	0.005700	0.028000

Initial State : Z 17

state	C(1)	C(2)	C(3)	C(4)
1	0.000095	0.000200	0.002730	0.012000
2	0.000005	0.000800	0.001170	0.048000
3	0.000095	0.000200	0.002730	0.016000
4	0.000005	0.000800	0.001170	0.064000
5	0.009310	0.019600	0.267540	0.012000
6	0.000490	0.078400	0.114660	0.048000
7	0.000190	0.001000	0.004200	0.036000
8	0.000010	0.004000	0.001800	0.144000
9	0.000190	0.001000	0.004200	0.048000
10	0.000010	0.004000	0.001800	0.192000
11	0.018620	0.098000	0.411600	0.036000
12	0.000980	0.392000	0.176400	0.144000
13	0.009215	0.000800	0.000070	0.012000
14	0.000485	0.003200	0.000030	0.048000
15	0.009215	0.000800	0.000070	0.016000
16	0.000485	0.003200	0.000030	0.064000
17	0.903070	0.078400	0.006860	0.012000
18	0.047530	0.313600	0.002940	0.048000

Initial State : Z 18

state	C(1)	C(2)	C(3)	C(4)
1	0.000001	0.000010	0.000195	0.018000
2	0.000099	0.000990	0.003705	0.042000
3	0.000001	0.000010	0.000195	0.024000
4	0.000099	0.000990	0.003705	0.056000
5	0.000098	0.000980	0.019110	0.018000
6	0.009702	0.097020	0.363090	0.042000
7	0.000002	0.000050	0.000300	0.054000
8	0.000198	0.004950	0.005700	0.126000
9	0.000002	0.000050	0.000300	0.072000
10	0.000198	0.004950	0.005700	0.168000
11	0.000196	0.004900	0.029400	0.054000
12	0.019404	0.485100	0.558600	0.126000
13	0.000097	0.000040	0.000005	0.018000
14	0.009603	0.003960	0.000095	0.042000
15	0.000097	0.000040	0.000005	0.024000
16	0.009603	0.003960	0.000095	0.056000
17	0.009506	0.003920	0.000490	0.018000
18	0.941094	0.388080	0.009310	0.042000

Goodness Measurements

State Z(i)	Combination val1	val2	val3	Goodness value	Remarks
1	0.00	0.00	0.00	0.00	
2	0.00	0.00	1.00	1.00	
3	0.00	0.50	0.00	0.50	
4	0.00	0.50	1.00	1.50	
5	0.00	1.00	0.00	1.00	
6	0.00	1.00	1.00	2.00	ideal
7	0.50	0.00	0.00	-0.50	
8	0.50	0.00	1.00	0.50	
9	0.50	0.50	0.00	0.00	
10	0.50	0.50	1.00	1.00	
11	0.50	1.00	0.00	0.50	
12	0.50	1.00	1.00	1.50	
13	1.00	0.00	0.00	-1.00	anti-ideal
14	1.00	0.00	1.00	0.00	
15	1.00	0.50	0.00	-0.50	
16	1.00	0.50	1.00	0.50	
17	1.00	1.00	0.00	0.00	
18	1.00	1.00	1.00	1.00	

Benefit Matrix

State Z(i)	C(1)	C(2)	C(3)	C(4)
1	0.055000	1.235000	0.340000	0.800000
2	-0.005000	0.425000	-0.010000	-0.300000
3	0.175000	1.180000	0.580000	0.400000
4	0.115000	0.370000	0.230000	-0.700000
5	0.020000	0.770000	0.270000	-0.100000
6	-0.040000	-0.040000	-0.080000	-1.200000
7	0.160000	1.595000	0.750000	1.200000
8	0.100000	0.785000	0.400000	0.100000
9	0.280000	1.540000	0.990000	0.800000
10	0.220000	0.730000	0.640000	-0.300000
11	0.125000	1.130000	0.680000	0.300000
12	0.065000	0.320000	0.330000	-0.800000
13	0.090000	1.600000	1.045000	1.700000
14	0.030000	0.790000	0.695000	0.600000
15	0.210000	1.545000	1.285000	1.300000
16	0.150000	0.735000	0.935000	0.200000
17	0.055000	1.135000	0.975000	0.800000
18	-0.005000	0.325000	0.625000	-0.300000

Decision Choices

State Z(i)	Choice Policy C(k)
1	2
2	4
3	2
4	2
5	4
6	1
7	2
8	2
9	2
10	1
11	2
12	3
13	4
14	2
15	2
16	3
17	2
18	1

set arbitary v(18) to 0

maxincome per period : 2.220446049E-15

APPENDIX D
UNIVERSITY SCHEDULE EXAMPLE

```
**********************************
**                              **
** Step I  : User   Interaction **
**                              **
**********************************
```

I. Formulate problem and prepare estimation probabilities

Problem description

P : Importance of problems to be solved
S : Degree of effectiveness in problem-solving
E : Potential energy of participants

p1=0 No important problem regarding teaching load, class size, and thesis advising

p2=.5 Moderate faculty shortage, large adjunct faculty, big class size, insufficient thesis advisors, shortage of required infrastructural facilities

p3=1 Acute shortage of faculty, immense class size, heavy thesis load, and conflicts in class scheduling

S1=0 Very poor quality teaching and research faculty, poor administration, inability to attract new faculty, uncontrolled student admission

S2=.5 Moderately effective faculty, reasonable student - faculty ratio, nominally effective administration

S3=1 Excellent quality teaching and research faculty, best student - faculty ratio, effective administration and leadership,ample supporting infrastructural facilities

E1=0 Faculty uninterested in teaching, insufficient time devoted to problem solving, low motivation

E2=1 High motivation, good productivity and commitment, dedicated teaching faculty, administrators

Alternative choices

C(1) Hire new faculty

C(2) Administer existing faculty better

C(3) Regulate student admission

C(4) Tailor class size to faculty availability

C(5) Add infrastructural facilities

	P			S			E	
Choice	0.00	0.50	1.00	0.00	0.50	1.00	0.00	1.00
1	0.94	0.74	0.05	0.10	0.05	0.01	0.67	0.08
1	0.05	0.25	0.30	0.80	0.35	0.20	0.33	0.92
1	0.01	0.01	0.65	0.10	0.60	0.79		
2	0.97	0.59	0.05	0.05	0.20	0.01	0.60	0.15
2	0.02	0.40	0.45	0.94	0.50	0.10	0.40	0.85
2	0.01	0.01	0.50	0.01	0.30	0.89		
3	0.96	0.60	0.01	0.40	0.05	0.01	0.91	0.08
3	0.03	0.38	0.40	0.55	0.30	0.01	0.09	0.92
3	0.01	0.02	0.59	0.05	0.65	0.98		
4	0.95	0.15	0.01	0.71	0.02	0.01	0.89	0.04
4	0.04	0.84	0.09	0.26	0.53	0.02	0.11	0.96
4	0.01	0.01	0.90	0.03	0.45	0.97		
5	0.90	0.24	0.08	0.84	0.13	0.11	0.92	0.30
5	0.05	0.75	0.40	0.12	0.57	0.43	0.08	0.70
5	0.05	0.01	0.52	0.04	0.30	0.46		

II. Execution

```
Select Menu Option
__________________
1. ExecGCM (Execute GCM Program)
2. ExecExit(Execution stop     )

Enter # of scale points for each factor :
__________________________________________
factor 1 :          3
factor 2 :          3
factor 3 :          2
goahead : Press any key, correction : "%"

Get Number of Choices for each State Z(i)    i = 1 .. 18
_________________________________________________________
State Z(i)          # of choices
__________          ____________
       1 ?                4
       2 ?                4
       3 ?                4
       4 ?                4
       5 ?                5
       6 ?                5
       7 ?                5
       8 ?                5
       9 ?                5
      10 ?                5
      11 ?                5
      12 ?                5
      13 ?                4
      14 ?                4
      15 ?                4
      16 ?                4
      17 ?                5
      18 ?                5
goahead : Press any key, correction : "%"
```

Get Estimation Probabilities

```
(factor1)
Initial  State f(1) =   0.00  Choice  C(1)
Terminal State :
 prob1  prob2  prob3 ?  0.94 0.05 0.01
Initial  State f(1) =   0.50  Choice  C(1)
Terminal State :
 prob1  prob2  prob3 ?  0.74 0.25 0.01
Initial  State f(1) =   1.00  Choice  C(1)
Terminal State :
 prob1  prob2  prob3 ?  0.05 0.30 0.65


         (* Display Input Data *)
            ____________________
              0.00    0.50    1.00
            ____________________
     1      0.94    0.74    0.05
     1      0.05    0.25    0.30
     1      0.01    0.01    0.65
goahead : Press any key, correction : "%"
 ..........................................
 ..........................................
 Repeat this step for remaining (n-1)factors
 ..........................................
 ..........................................
```

```
(**********************************)
(**                              **)
(** Step II : Generate  Matrices **)
(**                              **)
(**********************************)
```

Transition Matrix

Initial State : Z 1

state	C(1)	C(2)	C(3)	C(4)
1	0.062980	0.029100	0.349440	0.600305
2	0.031020	0.019400	0.034560	0.074195
3	0.503840	0.547080	0.480480	0.219830
4	0.248160	0.364720	0.047520	0.027170
5	0.062980	0.005820	0.043680	0.025365
6	0.031020	0.003880	0.004320	0.003135
7	0.003350	0.000600	0.010920	0.025276
8	0.001650	0.000400	0.001080	0.003124
9	0.026800	0.011280	0.015015	0.009256
10	0.013200	0.007520	0.001485	0.001144
11	0.003350	0.000120	0.001365	0.001068
12	0.001650	0.000080	0.000135	0.000132
13	0.000670	0.000300	0.003640	0.006319
14	0.000330	0.000200	0.000360	0.000781
15	0.005360	0.005640	0.005005	0.002314
16	0.002640	0.003760	0.000495	0.000286
17	0.000670	0.000060	0.000455	0.000267
18	0.000330	0.000040	0.000045	0.000033

Initial State : Z 2

state	C(1)	C(2)	C(3)	C(4)
1	0.007520	0.007275	0.030720	0.026980
2	0.086480	0.041225	0.353280	0.647520
3	0.060160	0.136770	0.042240	0.009880
4	0.691840	0.775030	0.485760	0.237120
5	0.007520	0.001455	0.003840	0.001140
6	0.086480	0.008245	0.044160	0.027360
7	0.000400	0.000150	0.000960	0.001136
8	0.004600	0.000850	0.011040	0.027264
9	0.003200	0.002820	0.001320	0.000416
10	0.036800	0.015980	0.015180	0.009984
11	0.000400	0.000030	0.000120	0.000048
12	0.004600	0.000170	0.001380	0.001152
13	0.000080	0.000075	0.000320	0.000284
14	0.000920	0.000425	0.003680	0.006816
15	0.000640	0.001410	0.000440	0.000104
16	0.007360	0.007990	0.005060	0.002496
17	0.000080	0.000015	0.000040	0.000012
18	0.000920	0.000085	0.000460	0.000288

Initial State : Z 3

state	C(1)	C(2)	C(3)	C(4)
1	0.031490	0.116400	0.043680	0.016910
2	0.015510	0.077600	0.004320	0.002090
3	0.220430	0.291000	0.262080	0.448115
4	0.108570	0.194000	0.025920	0.055385
5	0.377880	0.174600	0.567840	0.380475
6	0.186120	0.116400	0.056160	0.047025
7	0.001675	0.002400	0.001365	0.000712
8	0.000825	0.001600	0.000135	0.000088
9	0.011725	0.006000	0.008190	0.018868
10	0.005775	0.004000	0.000810	0.002332
11	0.020100	0.003600	0.017745	0.016020
12	0.009900	0.002400	0.001755	0.001980
13	0.000335	0.001200	0.000455	0.000178
14	0.000165	0.000800	0.000045	0.000022
15	0.002345	0.003000	0.002730	0.004717
16	0.001155	0.002000	0.000270	0.000583
17	0.004020	0.001800	0.005915	0.004005
18	0.001980	0.001200	0.000585	0.000495

Initial State : Z 4

state	C(1)	C(2)	C(3)	C(4)
1	0.003760	0.029100	0.003840	0.000760
2	0.043240	0.164900	0.044160	0.018240
3	0.026320	0.072750	0.023040	0.020140
4	0.302680	0.412250	0.264960	0.483360
5	0.045120	0.043650	0.049920	0.017100
6	0.518880	0.247350	0.574080	0.410400
7	0.000200	0.000600	0.000120	0.000032
8	0.002300	0.003400	0.001380	0.000768
9	0.001400	0.001500	0.000720	0.000848
10	0.016100	0.008500	0.008280	0.020352
11	0.002400	0.000900	0.001560	0.000720
12	0.027600	0.005100	0.017940	0.017280
13	0.000040	0.000300	0.000040	0.000008
14	0.000460	0.001700	0.000460	0.000192
15	0.000280	0.000750	0.000240	0.000212
16	0.003220	0.004250	0.002760	0.005088
17	0.000480	0.000450	0.000520	0.000180
18	0.005520	0.002550	0.005980	0.004320

Initial State : Z 5

state	C(1)	C(2)	C(3)	C(4)	C(5)
1	0.006298	0.005820	0.008736	0.008455	0.091080
2	0.003102	0.003880	0.000864	0.001045	0.007920
3	0.125960	0.058200	0.008736	0.016910	0.356040
4	0.062040	0.038800	0.000864	0.002090	0.030960
5	0.497542	0.517980	0.856128	0.820135	0.380880
6	0.245058	0.345320	0.084672	0.101365	0.033120
7	0.000335	0.000120	0.000273	0.000356	0.005060
8	0.000165	0.000080	0.000027	0.000044	0.000440
9	0.006700	0.001200	0.000273	0.000712	0.019780
10	0.003300	0.000800	0.000027	0.000088	0.001720
11	0.026465	0.010680	0.026754	0.034532	0.021160
12	0.013035	0.007120	0.002646	0.004268	0.001840
13	0.000067	0.000060	0.000091	0.000089	0.005060
14	0.000033	0.000040	0.000009	0.000011	0.000440
15	0.001340	0.000600	0.000091	0.000178	0.019780
16	0.000660	0.000400	0.000009	0.000022	0.001720
17	0.005293	0.005340	0.008918	0.008633	0.021160
18	0.002607	0.003560	0.000882	0.001067	0.001840

Initial State : Z 6

state	C(1)	C(2)	C(3)	C(4)	C(5)
1	0.000752	0.001455	0.000768	0.000380	0.029700
2	0.008648	0.008245	0.008832	0.009120	0.069300
3	0.015040	0.014550	0.000768	0.000760	0.116100
4	0.172960	0.082450	0.008832	0.018240	0.270900
5	0.059408	0.129495	0.075264	0.036860	0.124200
6	0.683192	0.733805	0.865536	0.884640	0.289800
7	0.000040	0.000030	0.000024	0.000016	0.001650
8	0.000460	0.000170	0.000276	0.000384	0.003850
9	0.000800	0.000300	0.000024	0.000032	0.006450
10	0.009200	0.001700	0.000276	0.000768	0.015050
11	0.003160	0.002670	0.002352	0.001552	0.006900
12	0.036340	0.015130	0.027048	0.037248	0.016100
13	0.000008	0.000015	0.000008	0.000004	0.001650
14	0.000092	0.000085	0.000092	0.000096	0.003850
15	0.000160	0.000150	0.000008	0.000008	0.006450
16	0.001840	0.000850	0.000092	0.000192	0.015050
17	0.000632	0.001335	0.000784	0.000388	0.006900
18	0.007268	0.007565	0.009016	0.009312	0.016100

Initial State : Z 7

state	C(1)	C(2)	C(3)	C(4)	C(5)
1	0.049580	0.017700	0.218400	0.094785	0.185472
2	0.024420	0.011800	0.021600	0.011715	0.016128
3	0.396640	0.332760	0.300300	0.034710	0.026496
4	0.195360	0.221840	0.029700	0.004290	0.002304
5	0.049580	0.003540	0.027300	0.004005	0.008832
6	0.024420	0.002360	0.002700	0.000495	0.000768
7	0.016750	0.012000	0.138320	0.530796	0.579600
8	0.008250	0.008000	0.013680	0.065604	0.050400
9	0.134000	0.225600	0.190190	0.194376	0.082800
10	0.066000	0.150400	0.018810	0.024024	0.007200
11	0.016750	0.002400	0.017290	0.022428	0.027600
12	0.008250	0.001600	0.001710	0.002772	0.002400
13	0.000670	0.000300	0.007280	0.006319	0.007728
14	0.000330	0.000200	0.000720	0.000781	0.000672
15	0.005360	0.005640	0.010010	0.002314	0.001104
16	0.002640	0.003760	0.000990	0.000286	0.000096
17	0.000670	0.000060	0.000910	0.000267	0.000368
18	0.000330	0.000040	0.000090	0.000033	0.000032

Initial State : Z 8

state	C(1)	C(2)	C(3)	C(4)	C(5)
1	0.005920	0.004425	0.019200	0.004260	0.060480
2	0.068080	0.025075	0.220800	0.102240	0.141120
3	0.047360	0.083190	0.026400	0.001560	0.008640
4	0.544640	0.471410	0.303600	0.037440	0.020160
5	0.005920	0.000885	0.002400	0.000180	0.002880
6	0.068080	0.005015	0.027600	0.004320	0.006720
7	0.002000	0.003000	0.012160	0.023856	0.189000
8	0.023000	0.017000	0.139840	0.572544	0.441000
9	0.016000	0.056400	0.016720	0.008736	0.027000
10	0.184000	0.319600	0.192280	0.209664	0.063000
11	0.002000	0.000600	0.001520	0.001008	0.009000
12	0.023000	0.003400	0.017480	0.024192	0.021000
13	0.000080	0.000075	0.000640	0.000284	0.002520
14	0.000920	0.000425	0.007360	0.006816	0.005880
15	0.000640	0.001410	0.000880	0.000104	0.000360
16	0.007360	0.007990	0.010120	0.002496	0.000840
17	0.000080	0.000015	0.000080	0.000012	0.000120
18	0.000920	0.000085	0.000920	0.000288	0.000280

Initial State : Z 9

state	C(1)	C(2)	C(3)	C(4)	C(5)
1	0.024790	0.070800	0.027300	0.002670	0.028704
2	0.012210	0.047200	0.002700	0.000330	0.002496
3	0.173530	0.177000	0.163800	0.070755	0.125856
4	0.085470	0.118000	0.016200	0.008745	0.010944
5	0.297480	0.106200	0.354900	0.060075	0.066240
6	0.146520	0.070800	0.035100	0.007425	0.005760
7	0.008375	0.048000	0.017290	0.014952	0.089700
8	0.004125	0.032000	0.001710	0.001848	0.007800
9	0.058625	0.120000	0.103740	0.396228	0.393300
10	0.028875	0.080000	0.010260	0.048972	0.034200
11	0.100500	0.072000	0.224770	0.336420	0.207000
12	0.049500	0.048000	0.022230	0.041580	0.018000
13	0.000335	0.001200	0.000910	0.000178	0.001196
14	0.000165	0.000800	0.000090	0.000022	0.000104
15	0.002345	0.003000	0.005460	0.004717	0.005244
16	0.001155	0.002000	0.000540	0.000583	0.000456
17	0.004020	0.001800	0.011830	0.004005	0.002760
18	0.001980	0.001200	0.001170	0.000495	0.000240

Initial State : Z 10

state	C(1)	C(2)	C(3)	C(4)	C(5)
1	0.002960	0.017700	0.002400	0.003350	0.001500
2	0.034040	0.100300	0.027600	0.001650	0.001000
3	0.020720	0.044250	0.014400	0.026800	0.028200
4	0.238280	0.250750	0.165600	0.013200	0.018800
5	0.035520	0.026550	0.031200	0.003350	0.000300
6	0.408480	0.150450	0.358800	0.001650	0.000200
7	0.001000	0.012000	0.001520	0.020100	0.013500
8	0.011500	0.068000	0.017480	0.009900	0.009000
9	0.007000	0.030000	0.009120	0.160800	0.253800
10	0.080500	0.170000	0.104880	0.079200	0.169200
11	0.012000	0.018000	0.019760	0.020100	0.002700
12	0.138000	0.102000	0.227240	0.009900	0.001800
13	0.000040	0.000300	0.000080	0.043550	0.015000
14	0.000460	0.001700	0.000920	0.021450	0.010000
15	0.000280	0.000750	0.000480	0.348400	0.282000
16	0.003220	0.004250	0.005520	0.171600	0.188000
17	0.000480	0.000450	0.001040	0.043550	0.003000
18	0.005520	0.002550	0.011960	0.021450	0.002000

Initial State : Z 11

state	C(1)	C(2)	C(3)	C(4)	C(5)
1	0.003640	0.006319	0.000400	0.000375	0.000320
2	0.000360	0.000781	0.004600	0.002125	0.003680
3	0.005005	0.002314	0.003200	0.007050	0.000440
4	0.000495	0.000286	0.036800	0.039950	0.005060
5	0.000455	0.000267	0.000400	0.000075	0.000040
6	0.000045	0.000033	0.004600	0.000425	0.000460
7	0.145600	0.056871	0.002400	0.003375	0.012800
8	0.014400	0.007029	0.027600	0.019125	0.147200
9	0.200200	0.020826	0.019200	0.063450	0.017600
10	0.019800	0.002574	0.220800	0.359550	0.202400
11	0.018200	0.002403	0.002400	0.000675	0.001600
12	0.001800	0.000297	0.027600	0.003825	0.018400
13	0.214760	0.568710	0.005200	0.003750	0.018880
14	0.021240	0.070290	0.059800	0.021250	0.217120
15	0.295295	0.208260	0.041600	0.070500	0.025960
16	0.029205	0.025740	0.478400	0.399500	0.298540
17	0.026845	0.024030	0.005200	0.000750	0.002360
18	0.002655	0.002970	0.059800	0.004250	0.027140

Initial State : Z 12

state	C(1)	C(2)	C(3)	C(4)	C(5)
1	0.000284	0.001675	0.006000	0.000455	0.000178
2	0.006816	0.000825	0.004000	0.000045	0.000022
3	0.000104	0.011725	0.015000	0.002730	0.004717
4	0.002496	0.005775	0.010000	0.000270	0.000583
5	0.000012	0.020100	0.009000	0.005915	0.004005
6	0.000288	0.009900	0.006000	0.000585	0.000495
7	0.002556	0.010050	0.054000	0.018200	0.001602
8	0.061344	0.004950	0.036000	0.001800	0.000198
9	0.000936	0.070350	0.135000	0.109200	0.042453
10	0.022464	0.034650	0.090000	0.010800	0.005247
11	0.000108	0.120600	0.081000	0.236600	0.036045
12	0.002592	0.059400	0.054000	0.023400	0.004455
13	0.025560	0.021775	0.060000	0.026845	0.016020
14	0.613440	0.010725	0.040000	0.002655	0.001980
15	0.009360	0.152425	0.150000	0.161070	0.424530
16	0.224640	0.075075	0.100000	0.015930	0.052470
17	0.001080	0.261300	0.090000	0.348985	0.360450
18	0.025920	0.128700	0.060000	0.034515	0.044550

Initial State : Z 13

state	C(1)	C(2)	C(3)	C(4)
1	0.003350	0.001500	0.003640	0.006319
2	0.001650	0.001000	0.000360	0.000781
3	0.026800	0.028200	0.005005	0.002314
4	0.013200	0.018800	0.000495	0.000286
5	0.003350	0.000300	0.000455	0.000267
6	0.001650	0.000200	0.000045	0.000033
7	0.020100	0.013500	0.145600	0.056871
8	0.009900	0.009000	0.014400	0.007029
9	0.160800	0.253800	0.200200	0.020826
10	0.079200	0.169200	0.019800	0.002574
11	0.020100	0.002700	0.018200	0.002403
12	0.009900	0.001800	0.001800	0.000297
13	0.043550	0.015000	0.214760	0.568710
14	0.021450	0.010000	0.021240	0.070290
15	0.348400	0.282000	0.295295	0.208260
16	0.171600	0.188000	0.029205	0.025740
17	0.043550	0.003000	0.026845	0.024030
18	0.021450	0.002000	0.002655	0.002970

Initial State : Z 14

state	C(1)	C(2)	C(3)	C(4)
1	0.000400	0.000375	0.000320	0.000284
2	0.004600	0.002125	0.003680	0.006816
3	0.003200	0.007050	0.000440	0.000104
4	0.036800	0.039950	0.005060	0.002496
5	0.000400	0.000075	0.000040	0.000012
6	0.004600	0.000425	0.000460	0.000288
7	0.002400	0.003375	0.012800	0.002556
8	0.027600	0.019125	0.147200	0.061344
9	0.019200	0.063450	0.017600	0.000936
10	0.220800	0.359550	0.202400	0.022464
11	0.002400	0.000675	0.001600	0.000108
12	0.027600	0.003825	0.018400	0.002592
13	0.005200	0.003750	0.018880	0.025560
14	0.059800	0.021250	0.217120	0.613440
15	0.041600	0.070500	0.025960	0.009360
16	0.478400	0.399500	0.298540	0.224640
17	0.005200	0.000750	0.002360	0.001080
18	0.059800	0.004250	0.027140	0.025920

Initial State : Z 15

state	C(1)	C(2)	C(3)	C(4)
1	0.001675	0.006000	0.000455	0.000178
2	0.000825	0.004000	0.000045	0.000022
3	0.011725	0.015000	0.002730	0.004717
4	0.005775	0.010000	0.000270	0.000583
5	0.020100	0.009000	0.005915	0.004005
6	0.009900	0.006000	0.000585	0.000495
7	0.010050	0.054000	0.018200	0.001602
8	0.004950	0.036000	0.001800	0.000198
9	0.070350	0.135000	0.109200	0.042453
10	0.034650	0.090000	0.010800	0.005247
11	0.120600	0.081000	0.236600	0.036045
12	0.059400	0.054000	0.023400	0.004455
13	0.021775	0.060000	0.026845	0.016020
14	0.010725	0.040000	0.002655	0.001980
15	0.152425	0.150000	0.161070	0.424530
16	0.075075	0.100000	0.015930	0.052470
17	0.261300	0.090000	0.348985	0.360450
18	0.128700	0.060000	0.034515	0.044550

Initial State : Z 16

state	C(1)	C(2)	C(3)	C(4)
1	0.000200	0.001500	0.000040	0.000008
2	0.002300	0.008500	0.000460	0.000192
3	0.001400	0.003750	0.000240	0.000212
4	0.016100	0.021250	0.002760	0.005088
5	0.002400	0.002250	0.000520	0.000180
6	0.027600	0.012750	0.005980	0.004320
7	0.001200	0.013500	0.001600	0.000072
8	0.013800	0.076500	0.018400	0.001728
9	0.008400	0.033750	0.009600	0.001908
10	0.096600	0.191250	0.110400	0.045792
11	0.014400	0.020250	0.020800	0.001620
12	0.165600	0.114750	0.239200	0.038880
13	0.002600	0.015000	0.002360	0.000720
14	0.029900	0.085000	0.027140	0.017280
15	0.018200	0.037500	0.014160	0.019080
16	0.209300	0.212500	0.162840	0.457920
17	0.031200	0.022500	0.030680	0.016200
18	0.358800	0.127500	0.352820	0.388800

Initial State : Z 17

state	C(1)	C(2)	C(3)	C(4)	C(5)
1	0.000335	0.000300	0.000091	0.000089	0.008096
2	0.000165	0.000200	0.000009	0.000011	0.000704
3	0.006700	0.003000	0.000091	0.000178	0.031648
4	0.003300	0.002000	0.000009	0.000022	0.002752
5	0.026465	0.026700	0.008918	0.008633	0.033856
6	0.013035	0.017800	0.000882	0.001067	0.002944
7	0.002010	0.002700	0.003640	0.000801	0.040480
8	0.000990	0.001800	0.000360	0.000099	0.003520
9	0.040200	0.027000	0.003640	0.001602	0.158240
10	0.019800	0.018000	0.000360	0.000198	0.013760
11	0.158790	0.240300	0.356720	0.077697	0.169280
12	0.078210	0.160200	0.035280	0.009603	0.014720
13	0.004355	0.003000	0.005369	0.008010	0.052624
14	0.002145	0.002000	0.000531	0.000990	0.004576
15	0.087100	0.030000	0.005369	0.016020	0.205712
16	0.042900	0.020000	0.000531	0.001980	0.017888
17	0.344045	0.267000	0.526162	0.776970	0.220064
18	0.169455	0.178000	0.052038	0.096030	0.019136

Initial State : Z 18

state	C(1)	C(2)	C(3)	C(4)	C(5)
1	0.000040	0.000075	0.000008	0.000004	0.002640
2	0.000460	0.000425	0.000092	0.000096	0.006160
3	0.000800	0.000750	0.000008	0.000008	0.010320
4	0.009200	0.004250	0.000092	0.000192	0.024080
5	0.003160	0.006675	0.000784	0.000388	0.011040
6	0.036340	0.037825	0.009016	0.009312	0.025760
7	0.000240	0.000675	0.000320	0.000036	0.013200
8	0.002760	0.003825	0.003680	0.000864	0.030800
9	0.004800	0.006750	0.000320	0.000072	0.051600
10	0.055200	0.038250	0.003680	0.001728	0.120400
11	0.018960	0.060075	0.031360	0.003492	0.055200
12	0.218040	0.340425	0.360640	0.083808	0.128800
13	0.000520	0.000750	0.000472	0.000360	0.017160
14	0.005980	0.004250	0.005428	0.008640	0.040040
15	0.010400	0.007500	0.000472	0.000720	0.067080
16	0.119600	0.042500	0.005428	0.017280	0.156520
17	0.041080	0.066750	0.046256	0.034920	0.071760
18	0.472420	0.378250	0.531944	0.838080	0.167440

Goodness Measurements

State Z(i)	Combination val1	val2	val3	Goodness value	Remarks
1	0.00	0.00	0.00	0.00	
2	0.00	0.00	1.00	1.00	
3	0.00	0.50	0.00	0.50	
4	0.00	0.50	1.00	1.50	
5	0.00	1.00	0.00	1.00	
6	0.00	1.00	1.00	2.00	ideal
7	0.50	0.00	0.00	-0.50	
8	0.50	0.00	1.00	0.50	
9	0.50	0.50	0.00	0.00	
10	0.50	0.50	1.00	1.00	
11	0.50	1.00	0.00	0.50	
12	0.50	1.00	1.00	1.50	
13	1.00	0.00	0.00	-1.00	anti-ideal
14	1.00	0.00	1.00	0.00	
15	1.00	0.50	0.00	-0.50	
16	1.00	0.50	1.00	0.50	
17	1.00	1.00	0.00	0.00	
18	1.00	1.00	1.00	1.00	

Benefit Matrix

State Z(i)	C(1)	C(2)	C(3)	C(4)	C(5)
1	0.795000	0.860000	0.390000	0.240000	
2	0.385000	0.310000	0.220000	0.090000	
3	0.570000	0.430000	0.365000	0.295000	
4	0.160000	-0.120000	0.195000	0.145000	
5	0.185000	0.320000	0.050000	0.060000	-0.320000
6	-0.225000	-0.230000	-0.120000	-0.090000	-0.700000
7	1.195000	1.170000	0.705000	0.340000	0.295000
8	0.785000	0.620000	0.535000	0.190000	-0.085000
9	0.970000	0.740000	0.680000	0.395000	0.280000
10	0.560000	0.190000	0.510000	-0.970000	-0.845000
11	-0.875000	-1.175000	0.120000	0.105000	-0.045000
12	-1.325000	-1.195000	-1.275000	-1.400000	-1.620000
13	1.030000	1.155000	0.625000	0.325000	
14	0.620000	0.605000	0.455000	0.175000	
15	0.805000	0.725000	0.600000	0.380000	
16	0.395000	0.175000	0.430000	0.230000	
17	0.420000	0.615000	0.285000	0.145000	0.035000
18	0.010000	0.065000	0.115000	-0.005000	-0.345000

Decision Choices

State Z(i)	Choice Policy C(k)
1	1
2	1
3	1
4	3
5	1
6	4
7	1
8	1
9	1
10	4
11	4
12	2
13	1
14	2
15	1
16	3
17	1
18	4

set arbitary v(18) to 0

maxincome per period : 5.412337245E-16

LIST OF REFERENCES

1. Tung Bui, Taracad R. Sivasankaran and Carson Eoyang *A Prescriptive Organizational Model of Garbage Can Choice Policies* Department of Administrative Sciences Naval Postgraduate School Monterey, CA 93943, Working Paper 86-13, 1986.

2. Cohen, M.D., J.G. March, and J.P. Olsen, *A Garbage Can Model of Organizational Choice*, Administrative Science Quarterly, 7, 1 (1972), pp. 1-25.

3. Popper, K., *The Logic of Scientific Discovery*, Hutchinson, London, 1959.

4. Dalkey, N., *An Elementary Cross-Impact Model*, Technological Forecasting Social Change, 3 (1972), pp. 341-351.

5. Kadane, J.B. and P.D. Larkey, *Subjective Probability and the Theory of Games*, Management Science, 28, 2(1982), pp. 113-120.

6. Howard, R.A., *Dynamic Programming and Markov Processes*, MIT Press, Cambridge, MA, 1960., pp. 32-69.

7. Taylor, H.M. and S. Karlin, *An Introduction to Stochastic Modeling*, Academic press, Orlando, FL, 1984, pp. 169-172.

8. Mokhtar S.Bazaraa and John J.Jarvis, *Linear Programming and Network Flows*, John Wiley and Sons, Inc., 1977, pp. 49,57.

9. Gordon B. Davis and Margrethe H. Olsen, *Management Information Systems* Mcgraw-Hill, Inc., 1985, pp. 164-169.

10. A. Arbel and R. M. Tong, *On the Generation of Alternatives in Dicision Analysis Problems*, Journal of Operational Research Society, 33:4, 1982, pp. 377-387.

11. Lawrence H. Miller *Advanced Programming : Design and Structure Using Pascal*, Addison-Wesley Publishing Company, Inc., 1986, pp. 101-118.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chief of Naval Operations Director, Information Systems (OP-945) Navy Department Washington, D.C. 20350-2000	1
4.	Department Chairman, code 52 Department of Computer Sciences Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Computer Technology Curricular Officer, code 37 Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Professor Sivasankaran, Taracad R., Code 54SJ Department of Administrative Sciences Naval Postgraduate School Monterey, CA 93943-5000	5
7.	CDR Gary S. Baker, Code 52BJ Department of Computer Sciences Naval Postgraduate School Monterey, CA 93943-5000	1
8.	Directorate of EDPS Department of Comptroller Staff Army Headquarters, 140-01, Seoul, Republic of Korea	1
9.	Library, P.O.Box 77 Postal Code 130-09 Gong Neung Dong, Dobong Goo Seoul, Republic of Korea	2

10. Major. Kang, Sun Mo 8
Postal Code 130-60
Kyung Gi Do,Yang Pyung Goon, Yang Pyung eub
Obin Ri, 233-11
Seoul, Republic of Korea

11. Kang, Sung Mo 1
1909 Trout Road Valley
Champaign, IL 61821

12. Maj. Choi, Seok Choel 1
1103 Sonoma # 3
Seaside, CA 93955

13. Maj. Kim, Dae Sik 1
1198 8th st. #2
Monterey, CA 93943

14. Maj. Yoon, Sang Il 1
1103 Sonoma # 4
Seaside, CA 93955

15. Capt, Yee, Seung Hee 1
1186 Pheonix # B
Seaside, CA 93955

16. Lt., Kim, Tae Woo 1
NPS BOQ #M-301
Monterey, CA 93943

Thesis
K14232 Kang
c.1
An interactive organizational choice processing system to support decision making by using a prescriptive garbage can model.